TECHNISCHE UNIVERSITÄT ILMENAU

DOCTORAL THESIS

Multi-Task Near-Field Perception for Autonomous Driving Using Surround-View Fisheye Cameras

Reviewers:
Prof. Dr.-Ing. Patrick Mäder
Prof. Dr. John Mc Donald
Prof. Dr. rer. nat. Gunther Notni

Author:
Dr.-Ing. Varun Ravi Kumar

A dissertation submitted in partial fulfillment
of the requirement for the degree of
Doctor of Engineering

in the

Fakultät für Informatik und Automatisierung
Softwaretechnik für sicherheitskritische Systeme

DOI: 10.22032/dbt.50751
URN: urn:nbn:de:gbv:ilm1-2021000421

Scientific Debate Day:
December 21, 2021

Declaration of Authorship

I, Dr.-Ing. Varun Ravi Kumar, declare that this thesis titled, "**Multi-Task Near-Field Perception for Autonomous Driving Using Surround-View Fisheye Cameras**" and the work presented in it are my own. I confirm that:

I certify that I prepared the submitted thesis independently without undue assistance of a third party and without the use of other than the indicated aids. Data and concepts directly or indirectly taken over from other sources have been marked stating the sources.

When selecting and evaluating the following materials, the persons listed below helped me in the way described respectively for a charge/free of charge:

- Senthil Yogamani
- Prof. Dr. John McDonald
- Marvin Klinger

Further persons were not involved in the content-material-related preparation of the thesis submitted. In particular, I have not used the assistance against payment offered by consultancies or placing services (doctoral consultants or other persons). I did not pay any money to persons directly or indirectly for work or services which are related to the content of the thesis submitted.

So far, the thesis has not been submitted identically or similarly to an examination office in Germany or abroad.

I have been notified that any incorrectness in the submitted declaration as mentioned above is assessed as an attempt to deceive and, according to § 7 para. 10 of the PhD regulations, this leads to a discontinuation of the doctoral procedure.

Signed: VarunR.

City, Date: Kronach, 05/01/2022

“You should not give up and we should not allow the problem to defeat us. To succeed in your mission, you must have single-minded devotion to your goal.”

Dr. A.P.J. Abdul Kalam

Dedicated to my parents Ravi Kumar and Vinutha, who have worked hard and sacrificed their lives for a better future for me and offered unconditional love support during my tough times. Thank you so much for everything!

Abstract

Die Bildung der Augen führte zum Urknall der Evolution. Die Dynamik änderte sich von einem primitiven Organismus, der auf den Kontakt mit der Nahrung wartete, zu einem Organismus, der durch visuelle Sensoren gesucht wurde. Das menschliche Auge ist eine der raffiniertesten Entwicklungen der Evolution, aber es hat immer noch Mängel. Der Mensch hat über Millionen von Jahren einen biologischen Wahrnehmungsalgorithmus entwickelt, der in der Lage ist, Autos zu fahren, Maschinen zu bedienen, Flugzeuge zu steuern und Schiffe zu navigieren. Die Automatisierung dieser Fähigkeiten für Computer ist entscheidend für verschiedene Anwendungen, darunter selbstfahrende Autos, Augmented Realität und architektonische Vermessung.

Die visuelle Nahfeldwahrnehmung im Kontext von selbstfahrenden Autos kann die Umgebung in einem Bereich von $0 - 10$ Metern und 360° Abdeckung um das Fahrzeug herum wahrnehmen. Sie ist eine entscheidende Entscheidungskomponente bei der Entwicklung eines sichereren automatisierten Fahrens. Jüngste Fortschritte im Bereich Computer Vision und Deep Learning in Verbindung mit hochwertigen Sensoren wie Kameras und LiDARs haben ausgereifte Lösungen für die visuelle Wahrnehmung hervorgebracht. Bisher stand die Fernfeldwahrnehmung im Vordergrund. Ein weiteres wichtiges Problem ist die begrenzte Rechenleistung, die für die Entwicklung von Echtzeit-Anwendungen zur Verfügung steht. Aufgrund dieses Engpasses kommt es häufig zu einem Kompromiss zwischen Leistung und Laufzeiteffizienz.

Wir konzentrieren uns auf die folgenden Themen, um diese anzugehen: 1) Entwicklung von Nahfeld-Wahrnehmungsalgorithmen mit hoher Leistung und geringer Rechenkomplexität für verschiedene visuelle Wahrnehmungsaufgaben wie geometrische und semantische Aufgaben unter Verwendung von faltbaren neuronalen Netzen. 2) Verwendung von Multi-Task-Learning zur Überwindung von Rechenengpässen durch die gemeinsame Nutzung von initialen Faltungsschichten zwischen den Aufgaben und die Entwicklung von Optimierungsstrategien, die die Aufgaben ausbalancieren.

Abstract

The formation of eyes led to the big bang of evolution. The dynamics changed from a primitive organism waiting for the food to come into contact for eating food being sought after by visual sensors. The human eye is one of the most sophisticated developments of evolution, but it still has defects. Humans have evolved a biological perception algorithm capable of driving cars, operating machinery, piloting aircraft, and navigating ships over millions of years. Automating these capabilities for computers is critical for various applications, including self-driving cars, augmented reality, and architectural surveying.

Near-field visual perception in the context of self-driving cars can perceive the environment in a range of $0 - 10$ meters and 360° coverage around the vehicle. It is a critical decision-making component in the development of safer automated driving. Recent advances in computer vision and deep learning, in conjunction with high-quality sensors such as cameras and LiDARs, have fueled mature visual perception solutions. Until now, far-field perception has been the primary focus. Another significant issue is the limited processing power available for developing real-time applications. Because of this bottleneck, there is frequently a trade-off between performance and run-time efficiency.

We concentrate on the following issues in order to address them: 1) Developing near-field perception algorithms with high performance and low computational complexity for various visual perception tasks such as geometric and semantic tasks using convolutional neural networks. 2) Using Multi-Task Learning to overcome computational bottlenecks by sharing initial convolutional layers between tasks and developing optimization strategies that balance tasks.

Publications

The research work presented in this thesis appears in the following publications, and the summarized contributions are solely from me. The following papers' chronology has been sorted on a top-down approach and descending order of the publication year.

Primary Authorship

1. Varun Ravi Kumar, Senthil Yogmani, Hazem Rashed, Ganesh Sitsu, Christian Witt, Isabelle Leang, Stefan Milz, Patrick Mäder. *"**OmniDet: Surround View Cameras based Multi-task Visual Perception Network for Autonomous Driving.**"* In the IEEE Robotics and Automation Letters (RA-L) + IEEE International Conference on Robotics and Automation (ICRA), 2021 [1].

   Google Site: `https://sites.google.com/view/omnidet/`

   Contribution:

   - Architect and lead developer of the first real-time six-task model for surround-view fisheye camera perception, out of which five were developed solely by me. For object detection, my contribution includes Open Neural Network Exchange (ONNX) model creation, setting up the training pipeline, and introducing a synergy between semantic segmentation and object detection.
   - Designed the multi-task learning framework in PyTorch with various state-of-the-art multi-task loss functions. It potentially replaced the current MTL framework in Tensorflow developed by many developers worldwide in Valeo, and the new framework helped to win next-generation projects.
   - Worked on creating the WoodScape dataset, including writing parsers and data loaders for each task. The dataset and the corresponding parsers will be released to the public on Github and will be part of the long-term maintenance.
   - To encourage further research in developing multi-task perception algorithms, the code was made public on the Github, which proved to be quite popular in the vision community and significantly helped the discernibility of the approach.
   - Integrated my novel approaches of distance estimation on fisheye camera images which are discussed in papers [2, 3, 4, 5, 6].
   - Modelled real-time capable, efficient network-architectures for semantic segmentation, motion segmentation, and soiling segmentation.
   - Performed embedded integration on NVIDIA's Jetson AGX with models exported to the ONNX format.

2. Varun Ravi Kumar, Ciarán Eising, , Christian Witt, Senthil Yogamani, Patrick Mäder. "***Surround-view Fisheye Camera Perception for Automated Driving: Overview, Survey & Challenges***" In-Review of the Journal of IEEE Transactions on Intelligent Transportation Systems, 2021.

   Contribution:

   - Summarized the perception tasks carried out in the thesis.

- Worked on describing the most commonly used fisheye camera projection models.

3. Varun Ravi Kumar, Marvin Klingner, Senthil Yogamani, Stefan Milz, Tim Fingscheidt, Patrick Mäder. *"**SynDistnet: Self-Supervised Monocular Fisheye Camera Distance Estimation Synergized with Semantic Segmentation for Autonomous Driving.**"* In the Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2021 [4].

   Oral Talk: `https://youtu.be/zL6zvtUy4cc`
   Google Site: `https://sites.google.com/view/syndistnet/`

   Contribution:

   - Solely developed a novel architecture to learn self-supervised distance estimation synergized with semantic segmentation and self-attention modules.
   - Research collaboration with Technische Universität Braunschweig to come up with a novel dynamic object filtering technique. Had fruitful discussions with Marvin Klinger (Ph.D. student) on depth estimation. The code and integration were carried out solely by me as Valeo's policy does not allow code sharing.
   - As a second contribution compared to PackNet [7], I came up with a one-stage training approach to filter dynamic objects by introducing synergy between distance estimation and semantic segmentation.

4. Varun Ravi Kumar, Marvin Klingner, Senthil Yogamani, Markus, Bach, Stefan Milz, Tim Fingscheidt, Patrick Mäder. *"**SVDistNet: Self-Supervised Near-Field Distance Estimation on Surround View Fisheye Cameras.**"* In the Journal of IEEE Transactions on Intelligent Transportation Systems.

   Google Site: `https://sites.google.com/view/svdistnet/`

   Contribution:

   - Designed and developed a novel camera geometry adaptive multi-scale convolution for fisheye camera models to enable the training and deployment of distance estimation models on multiple surround-view cameras mounted on different viewpoints.
   - With this design, I proposed a solution for surround-view fisheye cameras targeting large-scale industrial deployment. In other words, Valeo can target the design of a model for production that can be deployed in millions of vehicles having its own set of cameras.
   - Demonstrated and showcased a single trained model for 12 fisheye cameras, which achieves the equivalent result as an individual specialized model that overfits to a particular camera model.
   - Integrated and improved vector-based self-attention network module [8] compared to my previous work [4] into the framework.

5. Varun Ravi Kumar, Senthil Yogamani, Stefan Milz, Patrick Mäder. *"**FisheyeDistanceNet++: Self-Supervised Scale-Aware Distance Estimation with Improved Training Speed and Accuracy through Loss Function Optimization.**"* In the Electronic Imaging Autonomous Vehicles and Machines (EI-AVM), 2021 [6].

   Contribution:

- As an incremental paper to FisheyeDistanceNet [2] depicted the importance of using a general robust loss function instead of the L_1 loss. I carried out the development of code and experiments.
- Performed extensive ablation studies on the impact of using normalization techniques on the results and designed optimal network architecture.

6. Varun Ravi Kumar, Sandesh Athni Hiremath, Markus Bach, Stefan Milz, Christian Witt, Clément Pinard, Senthil Yogamani, Patrick Mäder. *"**FisheyeDistanceNet: Self-Supervised Scale-Aware Distance Estimation using Monocular Fisheye Camera for Autonomous Driving.**"* In the IEEE International Conference on Robotics and Automation (ICRA), 2020 [2].

   Oral Talk: `https://youtu.be/qAsdpHP5e8c`
   Google Site: `https://sites.google.com/view/fisheyedistancenet/`

   Contribution:

   - Formulated the novel self-supervised training strategy to infer a distance map from a sequence of distorted and unrectified raw fisheye camera images.
   - With the research guidance from Clément Pinard, I came up with a novel solution to the scale factor uncertainty with the bolster from ego-motion velocity that allows outputting metric distance maps. This facilitates the map's practical use for self-driving cars and validates the training of the CNN. This contribution is considered the heart of the entire thesis, as this is vital to enable the training of a CNN to estimate metric distance on raw fisheye camera images. A significant idea that laid a foundation for the following research in this domain.
   - Created a subset of the Woodscape dataset for distance estimation task, including recording the raw scenes and their post-processing. Wrote parsers and transformation scripts to obtain the ground-truth LiDAR samples to validate the network's estimates.
   - Incorporated my novel idea from [9, 10] to solve the problem of occlusion due to different mounting positions of LiDAR and camera.
   - Solely designed and developed novel ideas and network architecture to obtain state-of-the-art results on the KITTI [11] and the WoodScape [12] dataset.

7. Varun Ravi Kumar, Senthil Yogamani, Markus Bach, Christian Witt, Stefan Milz, Patrick Mäder. *"**UnRectDepthnet: Self-supervised Monocular Depth Estimation using a Generic Framework for Handling Common Camera Distortion Models.**"* In the International Conference on Intelligent Robots and Systems (IROS), 2020 [3].

   Oral Talk: `https://youtu.be/3Br2KSWZRrY`
   Google Site: `https://sites.google.com/view/unrectdepthnet/`

   Contribution:

   - Following upon the success of [2], created a novel generic end-to-end self-supervised training pipeline to estimate monocular depth maps on raw distorted images for various camera models.

- Demonstrated the first results of depth estimation directly on unrectified KITTI sequences and achieved state-of-the-art results on the KITTI depth estimation dataset among self-supervised methods.

8. Varun Ravi Kumar, Stefan Milz, Christian Witt, Martin Simon, Karl Amende, Johannes Petzold, Senthil Yogamani, Timo Pech. ***"Near-field Depth Estimation using Monocular fisheye camera: A Semi-Supervised Learning Approach using Sparse LiDAR Data."*** In the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2018 [9].

   Contribution:

   - Built a joint fisheye camera and LiDAR dataset for supervised training of distance estimation.
   - Adapted the training data to handle occlusions due to differences between camera and LiDAR viewpoint with a novel occlusion correction algorithm.

9. Varun Ravi Kumar, Stefan Milz, Christian Witt, Martin Simon, Karl Amende, Johannes Petzold, Senthil Yogamani, Timo Pech. ***"Monocular Fisheye Camera Depth Estimation using Sparse LiDAR Supervision."*** In the 21st International Conference on Intelligent Transportation Systems (ITSC), 2018 [10].

   Google Site: `https://sites.google.com/view/FisheyeLiDARDepth/`

   Contribution:

   - Developed the first preliminary supervised approach work before the self-supervised FisheyeDistanceNet [2], to demonstrate fisheye camera distance estimation using CNN.
   - Demonstrated a working prototype purely trained on sparse Velodyne LiDAR data.
   - Tailored the loss function and training algorithm to handle sparse-depth data.

Secondary Authorship

1. Michal Uřičář, Ganesh Sistu, Hazem Rashed, Antonín Vobecký, Varun Ravi Kumar, Pavel Křížek, Fabian Bürger, Senthil Yogamani. ***"Let's Get Dirty: GAN Based Data Augmentation for Camera Lens Soiling Detection in Autonomous Driving."*** In the Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2021 [13].

   Contribution:

   - Part of the research discussions and reviewed the code and oral presentation.
   - Supported the dataset integration into the model.

2. Hazem Rashed, Eslam Mohamed, Ganesh Sistu, Varun Ravi Kumar, Ciarán Eising, Ahmad El-Sallab, Senthil Yogamani. ***"Generalized Object Detection on Fisheye Cameras for Autonomous Driving: Dataset, Representations and Baseline."*** In the Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2021 [14].

   Google Site: `https://sites.google.com/view/fisheyeyolo/`

Contribution:

- Major contribution in writing the research paper.
- Designed the encoder for object detection and was part of the development of the YOLOv3 decoder.
- Integrated different representations of the bounding box to the framework and trained the standard bounding box representation.
- Active member of the research discussions and converted the model to ONNX to run it on an automotive embedded platform.

3. Ibrahim Sobh, Ahmed Hamed, Varun Ravi Kumar, and Senthil Yogamani. *"Adversarial Attacks on Multi-Task Visual Perception for Autonomous Driving"* Journal of Imaging Science and Technology (JIST), 2021 [15].

   Contribution:

   - Major contribution in writing the research paper and brainstorming the adversarial attacks.
   - Provided the inference and model boiler-plate code for white-box and black-box attacks from the *OmniDet* framework.

4. Sebastian Houben, Stephanie Abrecht, Maram Akila, Andreas Bär, Felix Brockherde, Patrick Feifel, Tim Fingscheidt, Sujan Sai Gannamaneni, Seyed Eghbal Ghobadi, Ahmed Hammam, Anselm Haselhoff, Felix Hauser, Christian Heinzemann, Marco Hoffmann, Nikhil Kapoor, Falk Kappel, Marvin Klingner, Jan Kronenberger, Fabian Küppers, Jonas Löhdefink, Michael Mlynarski, Michael Mock, Firas Mualla, Svetlana Pavlitskaya, Maximilian Poretschkin, Alexander Pohl, Varun Ravi Kumar, Julia Rosenzweig, Matthias Rottmann, Stefan Rüping, Timo Sämann, Jan David Schneider, Elena Schulz, Gesina Schwalbe, Joachim Sicking, Toshika Srivastava, Serin Varghese, Michael Weber, Sebastian Wirkert, Tim Wirtz and Matthias Woehrle. *"Inspect, Understand, Overcome: A Survey of Practical Methods for AI Safety"* arXiv preprint arXiv:2104.14235 [16].

   Contribution:

   - Contributed the summary on Multi Task Learning.

5. Ashok Dahal, Varun Ravi Kumar, Senthil Yogamani and Ciarán Eising. *"An Online Learning System for Wireless Charging Alignment using Surround-view Fisheye Cameras."* In the IEEE Transactions on Intelligent Transportation Systems, 2021.

   Contribution:

   - Major contribution in writing the research paper and brainstorming the idea.
   - Provided the inference and model boiler-plate code from *OmniDet* framework.

6. Mahesh M Dhananjaya, Varun Ravi Kumar and Senthil Yogamani. *"Weather and Light Level Classification for Autonomous Driving: Dataset, Baseline and Active Learning"* In-Review of the 24th International Conference on Intelligent Transportation Systems (ITSC), 2021 [17].

   Contribution:

- Major contribution in writing the research paper and mentored the intern during his master thesis.
- Provided the training and model boiler-plate code for the weather classification task from the *OmniDet* framework.

7. Ashok Dahal, Eric Golab, Rajender Garlapati, Varun Ravi Kumar, Senthil Yogamani. ***"RoadEdgeNet: Road Edge Detection System Using Surround View Camera Images."*** In the Electronic Imaging Autonomous Vehicles and Machines (EI-AVM), 2021 [18].

   Contribution:

   - Part of the research discussions, reviewed the code and contribution in writing the research paper.

8. Hazem Rashed, Eslam Mohamed, Ganesh Sistu, Varun Ravi Kumar, Ciarán Eising, Ahmad El-Sallab, Senthil Yogamani. ***"FisheyeYOLO: Object Detection on Fisheye Cameras for Autonomous Driving."*** In the NeurIPS Workshop on Machine Learning for Autonomous Driving, 2021 [19].

   Contribution:

   - Major contribution in writing the short research paper derived from [14].

9. Arindam Das, Pavel Křížek, Ganesh Sistu, Fabian Bürger, Sankaralingam Madasamy, Michal Uřičář, Varun Ravi Kumar, Senthil Yogamani. ***"TiledSoilingNet: Tile-level Soiling Detection on Automotive Surround-view Cameras Using Coverage Metric."*** In the IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), 2020 [20].

   Contribution:

   - Part of the research discussions and reviewed the code and oral presentation.

10. Marie Yahiaoui, Hazem Rashed, Letizia Mariotti, Ganesh Sistu, Ian Clancy, Lucie Yahiaoui, Varun Ravi Kumar, Senthil Yogamani. ***"FisheyeModNet: Moving Object Detection on Surround-View Cameras for Autonomous Driving."*** In the International Conference on Computer Vision (ICCV) Workshop on 360° Perception and Interaction, 2019 [21].

    Contribution:

    - Re-implemented the entire approach with novel improvements in PyTorch [22] and obtained better results than the former.
    - Part of the original approach and discussions.

Publications

Primary Authorship

1. V. Ravi Kumar, S. Yogamani, H. Rashed, G. Sitsu, C. Witt, I. Leang, S. Milz,and P. Mäder, "OmniDet: Surround View Cameras based Multi-task VisualPerception Network for Autonomous Driving," in *IEEE Robotics and Automation Letters (RA-L) + 2021 IEEE International Conference on Robotics and Automation (ICRA),* vol. 6, no. 2, 2021, pp. 2830–2837.

2. V. Ravi Kumar, C. Witt, S. Yogamani, and P. Mäder, "Surround-View Fisheye Camera Perception for Automated Driving: Overview, Survey & Challenges," in *Review of the Journal of IEEE Transactions on Intelligent Transportation Systems, 2021.*

3. V. Ravi Kumar, M. Klingner, S. Yogamani, M. Bach, S. Milz, T. Fingscheidt, and P. Mäder, "SVDistNet: Self-Supervised Near-Field Distance Estimation on Surround View Fisheye Cameras," *IEEE Transactions on Intelligent Transportation Systems, vol. abs/2104.04420, 2021.*

4. V. Ravi Kumar, M. Klingner, S. Yogamani, S. Milz, T. Fingscheidt, and P. Mader, "Syndistnet: Self-supervised monocular fisheye camera distance estimation synergized with semantic segmentation for autonomous driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021*, pp. 61–71.

5. V. Ravi Kumar, S. Yogamani, S. Milz, and P. Mäder, "FisheyeDistanceNet++: Self-Supervised Fisheye Distance Estimation with Self-Attention, Robust Loss Function and Camera View Generalization," in *Electronic Imaging. Society for Imaging Science and Technology, 2021.*

6. V. Ravi Kumar, S. A. Hiremath, M. Bach, S. Milz, C. Witt, C. Pinard, S. Yogamani, and P. Mäder, "Fisheyedistancenet: Self-supervised scale-aware distance estimation using monocular fisheye camera for autonomous driving," in *2020 IEEE International Conference on Robotics and Automation (ICRA), 2020*, pp. 574–581

7. V. Ravi Kumar, S. Yogamani, M. Bach, C. Witt, S. Milz, and P. Mäder, "UnRectDepthNet: Self-Supervised Monocular Depth Estimation using a Generic Framework for Handling Common Camera Distortion Models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS) 2020*, pp. 8177–8183.

8. V. Ravi Kumar, S. Milz, C. Witt, M. Simon, K. Amende, J. Petzold, S. Yogamani, and T. Pech, "Monocular fisheye camera depth estimation using sparse lidar supervision," in *21st International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 2853–2858.*

9. V. Ravi Kumar, S. Milz, C. Witt, M. Simon, K. Amende, J. Petzold, S. Yogamani, and T. Pech, "Near-field depth estimation using monocular fisheye camera A semi-supervised learning approach using sparse LiDAR data," in *CVPR Workshop, vol. 7, 2018.*

Secondary Authorship

1. M. Uřičář, G. Sistu, H. Rashed, A. Vobecký, V. Ravi Kumar, P. Křížek, F. Bürger, and S. Yogamani, "Let's Get Dirty: GAN Based Data Augmentation for Camera Lens Soiling Detection in Autonomous Driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 766–775.*

2. Gallagher, Louis and Ravi Kumar, Varun and Yogamani, Senthil and McDonald, John ,B "A Hybrid Sparse-Dense Monocular SLAM System for Autonomous Driving," in *2021 European Conference on Mobile Robots (ECMR), 2021, pp. 1–8.*

3. H. Rashed, E. Mohamed, G. Sistu, V. Ravi Kumar, C. Eising, A. El-Sallab, and S. Yogamani, "Generalized Object Detection on Fisheye Cameras for Autonomous Driving: Dataset, Representations and Baseline," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 2272–2280.*

4. I. Sobh, A. Hamed, V. Ravi Kumar, and S. Yogamani, "Adversarial Attacks on Multi-task Visual Perception for Autonomous Driving," in *Review of the IEEE 24th International Conference on Intelligent Transportation Systems (ITSC).*

5. A. Dahal, V. Ravi Kumar, S. Yogamani and C. Eising. "An Online Learning System for Wireless Charging Alignment using Surround-view Fisheye Cameras." *In the IEEE Transactions on Intelligent Transportation Systems, 2021.*

6. Sebastian Houben, Stephanie Abrecht, Maram Akila, Andreas Bär, Felix Brockherde, Patrick Feifel, Tim Fingscheidt, Sujan Sai Gannamaneni, Seyed Eghbal Ghobadi, Ahmed Hammam, Anselm Haselhoff, Felix Hauser, Christian Heinzemann, Marco Hoffmann, Nikhil Kapoor, Falk Kappel, Marvin Klingner, Jan Kronenberger, Fabian Küppers, Jonas Löhdefink, Michael Mlynarski, Michael Mock, Firas Mualla, Svetlana Pavlitskaya, Maximilian Poretschkin, Alexander Pohl, Varun Ravi Kumar, Julia Rosenzweig, Matthias Rottmann, Stefan Rüping, Timo Sämann, Jan David Schneider, Elena Schulz, Gesina Schwalbe, Joachim Sicking, Toshika Srivastava, Serin Varghese, Michael Weber, Sebastian Wirkert, Tim Wirtz and Matthias Woehrle. "Inspect, Understand, Overcome: A Survey of Practical Methods for AI Safety," *arXiv preprint arXiv:2104.14235, 2021.*

7. M. M. Dhananjaya, V. R. Kumar, and S. Yogamani, "Weather and Light Level Classification for Autonomous Driving: Dataset, Baseline and Active Learning," in *Review of the IEEE 24th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2021.*

8. A. Dahal, E. Golab, R. Garlapati, V. Ravi Kumar, and S. Yogamani, "RoadEdgeNet: Road Edge Detection System Using Surround View Camera Images," in *Electronic Imaging. Society for Imaging Science and Technology, 2021.*

9. A. Das, P. Křížek, G. Sistu, F. Bürger, S. Madasamy, M. Uřičář, V. Ravi Kumar, and S. Yogamani, "TiledSoilingNet: Tile-level Soiling Detection on Automotive Surround-view Cameras Using Coverage Metric," in *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2020, pp.1–6.*

10. H. Rashed, E. Mohamed, G. Sistu, V. Ravi Kumar, C. Eising, A. El-Sallab, and S. Yogamani, "FisheyeYOLO: Object Detection on Fisheye Cameras for Autonomous Driving," *Machine Learning for Autonomous Driving NeurIPS 2020 Virtual Workshop, 2020.*

11. M. Yahiaoui, H. Rashed, L. Mariotti, G. Sistu, I. Clancy, L. Yahiaoui, V. Ravi Kumar, and S. Yogamani, “FisheyeModNet: Moving object detection on Surround-View Cameras for Autonomous Driving,” *arXiv preprint arXiv:1908.11789, 2019*.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my Ph.D. advisor *Prof. Dr.-Ing. Patrick Mäder* for the continuous support of my Ph.D. study and research, for his patience, motivation, enthusiasm, and immense knowledge.

I am very grateful to *Senthil Yogamani* from Valeo, Ireland, who ensured that all my work became publications. I am so lucky to get to work with someone who inspires me every day. My sincere thanks to him for his guidance and leadership. His advice helped me in research, writing this thesis, and research papers. I could not have imagined having a better mentor for my thesis study who believed in me during my entire work. I thank him for his kindness and valuable comments and discussions to complete work on time. Apart from the research talks, he lead me throughout the thesis during my tough times and motivated me to reach my goal in the end. Finally, thanks for being available 24/7 as a mentor and a friend.

There are many people from Valeo whom I would like to thank, whose contributions have helped me make this thesis possible. I would like to express my gratitude to my supervisor *Dr. Stefan Milz* from Valeo, for the useful comments, remarks, and engagement throughout the Ph.D. thesis's learning process. *Johannes Petzold*, my manager, and all my colleagues at Valeo, Kronach, for providing me with a wonderful environment to work in. Grateful to *Clément Pinnard* for the initial research discussions on the *SfM* framework and the discussions about depth estimation. Special thanks to *Christian Witt* for teaching me advanced python and not so joyful rigorous code reviews and always being available for research discussions. *Markus Bach* for all the math fun and research ideas, funny conversations about random things, and always being willing to help me. *Kai Fischer*, for all the fun talks about the work and food, arguments about Python vs. C++. *Sandesh Hiremath* for the initial research discussions. *Martin Simon*, my team, lead for encouraging me at times during my thesis. *Ganesh Sitsu* for all the awesome fun chats and the WoodScape dataset creation. *Hazem Rashed* for all the hard work during MTL integration. *Isabelle Leang* and *Fabian Bürger* for the research discussions on MTL. *Michal Uřičář* for the discussions on GAN's. *Marvin Klinger* from TU Braunschweig, for the research collaboration, interminable paper corrections, and writing. To all the other people who helped me during my Ph.D. time.

I would like to thank my friends and family. My parents, especially my mother, for being present for me throughout the years. Their prayers, sacrifices for educating me and preparing me for the future. They have been a constant support; as a result of which, I could pursue a Master's and a Ph.D. degree in Germany. I would like to thank my close friends *Bharath Krishnaiah* and *Jagadish Subramani*, who were part of my Master's cohort, currently working in Germany, for being supportive throughout my Ph.D. during my tough times and during the corona crisis. Thanks for all the food and memorable, joyful moments in life. To *Pratik Kamble* for making sure I didn't sleep enough, for making me believe in myself, and for listening to all my research ideas late at night. To my childhood friend *Sriram*, for being part of my entire educational journey from high school to engineering and reviewing my literature. Finally, I would like to express my thanks to those involved directly or indirectly in completing my project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

Since the early 1960s, we have been pursuing the fantasy of commuting between places while sitting in a driverless car with no manual intervention. Over the last decade, autonomous driving (AD) has piqued the interest of vehicle manufacturers more than ever before. The vast and ground-breaking advances in artificial intelligence (AI) and computer vision made possible by machine learning are the primary drivers of this developing trend.

Let us consider the case of an automobile. According to global statistics, approximately **3700** lives are lost due to road accidents every day (approximately **1.35 million** people per year) and **20** and **50 million** people are left with non-fatal injuries [23]. Out of those accidents, more than 70% are caused due to human errors. Despite robust safety standards developed by the manufacturers and technology evolving massively, we have not reached an acceptable number of traffic accidents. What could be the possible reason? Do we have a long-term solution for this? Indeed, AI and autonomous systems could work as magic bullets in these situations. So, the basic principle involves machines taking control over everything. This would mean eradicating human interventions completely, which is the root cause of many of these problems.

An autonomous vehicle drives itself without the assistance of a human operator, using a collection of sensors, cameras, radar, and AI algorithms. Experts have identified five stages in the growth of self-driving vehicles. Each level defines how much a car may take over activities and obligations from its driver, as well as how the car and driver interact: 1) Driver assistance, 2) Semi-automated driving, 3) Highly automated driving, and 4) Fully automated driving 5) Complete automation [24]. The complete AD system can be roughly encompassed in five primary components across all five levels, as indicated in Figure 1.1.

Perception System in Action: Sensors collect data from the environment and send it to the next ring of this chain, *Environment Perception* where it is processed for the subsequent decision-making processes regarding the vehicle's next action. As a result, the extracted information at this point is critical for an autonomous vehicle. There are different types of sensors available to collect data from the environment. Our attention is drawn to image sensors, which convert light waves into signals that transfer information to build an image. Each camera is made up primarily of lenses

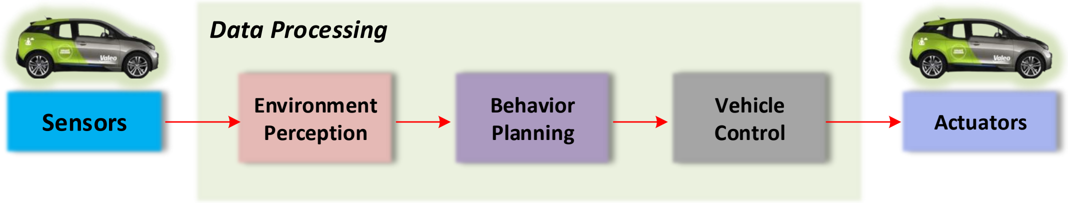


Figure 1.1 **Major components of an autonomous driving system [26].**

and sensors. Most autonomous vehicle industry efforts are focused on advanced driver assistance systems (ADAS) since that is the first step for fully self-driving cars. The backbone of all ADAS applications is cameras, usually monocular or stereo vision systems. Regardless of the method, cameras are the basis for safe autonomous cars that can "*see and drive*" themselves. Features, such as adaptive cruise control, can be implemented robustly as a fusion of radar or LiDAR data with cameras, usually for non-curvy roads and higher speeds. At present, this method has guided autonomous vehicles and ADAS to **Level 2** autonomy. In these situations, vehicles can control certain functions only — like emergency braking and advanced lane assist, and lane change. Still, in the end, humans are behind the wheel of these vehicles [25].

The holistic scene understanding of the environment makes autonomous cars possible. Without this ability to sense the near-field environment, the autonomous systems would not have a way to know what speed to set, decide when to turn, when to make a lane switch or when to apply brakes. The complex, intelligent systems should have these abilities equipped to make split-second life-saving decisions. When an autonomous vehicle travels from source to destination, a navigator such as Google Maps or high-definition maps generates a high-level route. A series of connected nodes at finite distances make up this route. The vehicle moves from one node to another in a repetitive way until it reaches its destination. The sensing stage collects information about the surroundings using one of three sensors or a combination of these sensors such as cameras, radar, and LiDAR. Cameras are one of the most traditional means since they can visually perceive the scene in the same way humans do with vision. Radars are used in conjunction with cameras as an auxiliary method to detect large objects. LiDAR uses an array of light pulses to measure distance. Some use a combination of all three sensors. Perception involves the extraction of useful information from the raw data like lane positions, pedestrians, and other vehicles [27], moving objects detection [28] and recognition of drivable regions. Localization is the vehicle's ability to precisely know its position in the real world at decimeter accuracy [29, 30]. In simple words, perception answers what is around the vehicle, and localization answers precisely where the car is. Path planning algorithms [31] make use of this related information to define a path to navigate from one node to another.

Historically, most autonomous car companies (*e.g.*, BMW, Audi, Toyota) have relied heavily on LiDAR since, until recently, neural networks were not powerful enough to handle multiple camera inputs. The laser sensors currently used to detect 3D objects in autonomous cars' paths are bulky, ugly, expensive, energy-inefficient – and highly accurate [32]. These LiDAR sensors are affixed to cars' roofs, increasing wind drag, a particular disadvantage for electric cars. They can add around $10,000\$$ to a car's cost. However, despite their drawbacks, most experts have considered LiDAR sensors the only plausible way for self-driving vehicles to safely perceive

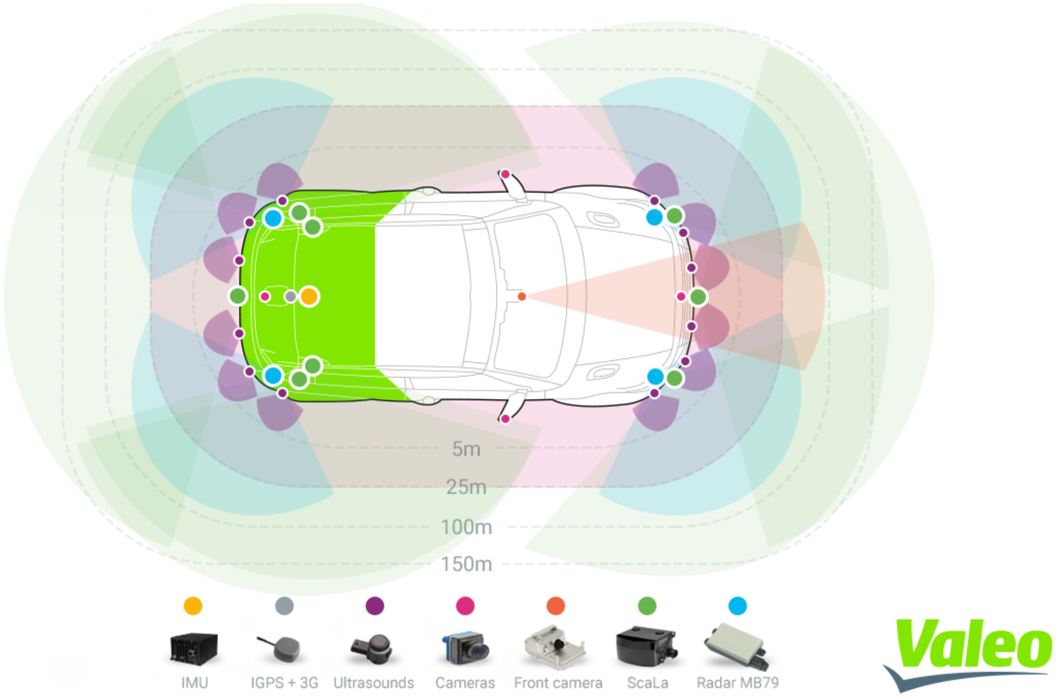


Figure 1.2 **Illustration of the near-field range and sensor suite of an exemplary self-driving car.**

pedestrians, cars, and other hazards on the road [32]. As with human eyes, cameras capture the resolution, small details, and vividness of a scene with such detail that no other sensors, including radar, ultrasonic, and lasers, can match [25]. *Tesla* is one of the most notable companies that has placed a big bet on cameras, integrating eight of them into each vehicle [33], along with a powerful deep neural network called HydraNets [34]. With the primary factor of cost and the inspiration from humans to mimic the way nature forged us to drive. *This thesis will mainly focus on building a unified perception system from one of the three primary sensors i.e. using cameras only*.

1.1 Goal

Near Field perception for AD is a region from 0-10 meters and 360° coverage around the vehicle as shown in Figure 1.2. Some of its use cases are automated parking, traffic jam assist, and urban driving. The sensor suite includes ultrasonics, fisheye-cameras, and radar (see 1.2). There are limited datasets and very little work on near-field perception tasks as the main focus is on far-field perception. In contrast to far-field, it is more challenging due to high precision object detection requirements of 10 cm. For example, let us look at the parking scenario in Figure 1.4. The car needs to be parked in a tight space with partial object visibility and no room for error, requiring high precision. Four fisheye cameras are sufficient to cover the near-field perception as shown in Figure 1.3.

Standard algorithms can not be extended easily on fisheye cameras due to their large radial distortion. There is very little work on the perception algorithms on fisheye cameras. Also, most of the current AD systems are **Level 2**. In this thesis, we focus on building a holistic 360° scene understanding for a near-field perception system that constitutes the necessary modules for a **Level 3** AD stack ***using four fisheye cameras***.

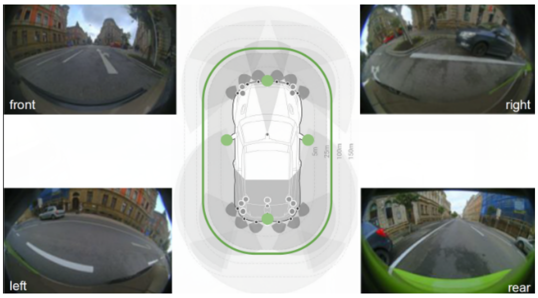


Figure 1.3 **Surround view system with four fisheye cameras.**

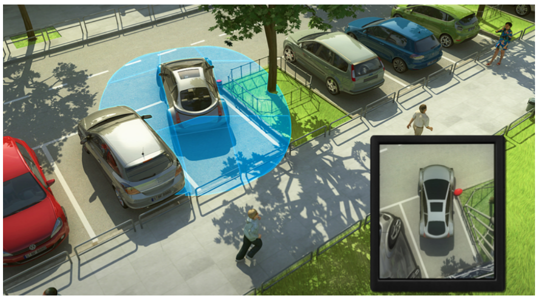


Figure 1.4 **Illustration of a tight parking scenario.** Figure reproduced from [35].

The developed framework will be called *OmniDet* for two reasons. Firstly, *Omni* can be used for wide-angle omnidirectional cameras. Secondly, the word *Omni* means all, and we are detecting all necessary objects for such a **Level 3** AD system. Thus, it is a framework that does a 360° Near-Field Detection.

The naive approach is rectifying the fisheye images and applying these algorithms. The standard question which arises when we talk about distortion in fisheye cameras is: *Why do we not rectify the images?*

- Most algorithms are usually designed to work on rectified pinhole camera images.
- Removing distortion leads to a significant loss in the Field-of-View.
- For a horizontal Field-of-View (hFoV) greater than 180°, rays incident from behind the camera make it theoretically impossible to establish a complete mapping to a rectilinear viewport. Thus the rectification defeats the purpose of using a wide-angle fisheye lens.
- Resampling distortion artifacts are particularly strong in the periphery as a small region in the fisheye image is expanded to a larger region in the rectified image. The texture is lost, and noise is introduced.

A few of the most important questions and challenges that arise for near field perception are answered by these perception tasks:

- What is the geometry of the scene? Furthermore, How far is an object at a pixel level in an image? **Monocular depth estimation**
- What is around me? and What type of object at a pixel level in an image? **Semantic Segmentation**
- How are different parts of the scene moving? or Is the object moving? **Motion Segmentation**
- How to identify objects and locate them? **2D Object Detection**
- Can we perceive the world around us clearly? **Soiling Detection**

What if a simple sensor modality can provide all the cues listed above? *A single **RGB Fisheye Camera***. One of the other primary goals of this thesis is to target large-scale industrial deployment of these tasks in millions of cars; to achieve it, we need to consider some critical practical factors. The model needs to be robust to intrinsic camera variations arising due to manufacturing tolerances. Low-cost embedded hardware is needed for commercial reasons. Henceforth, the performance of these models needs to be real-time capable for driving scenarios. Model validation and safety certifications are required for compliance. Considering all this into account, we propose to use a single **multi-task learning (MTL)** (learn multiple tasks using a single input) model in contrast to using multiple models for different tasks. The MTL framework would be capable of learning geometry and semantics-related tasks from ***monocular videos only***. To further analyze the network's vulnerability against adversarial attacks, we plan to apply white and black box attacks for targeted and untargeted cases while attacking a task and inspecting the effect on all the others. The most challenging perception problem of all is the *distance estimation* on raw fisheye cameras. We will explore this geometry task in detail and create novel methodologies to obtain distance maps on the fisheye camera. We also perform the first detailed study on object detection on fisheye cameras and explore various better representations of bounding boxes to adapt the fisheye camera's geometry. We also introduce synergy between depth, semantics, and object detection afterward and complete the MTL framework.

1.2 Outline

At first, we will look into the basics of the commonly used fisheye camera projection models in **Chapter** 2 followed by the basics and definition of the perception tasks. In **Chapter** 3 we will look into the background of all the perception tasks. In this thesis, we will focus mainly on the distance estimation on raw fisheye monocular videos in **Chapter** 4 and **Chapter** 5. **Chapter** 4 shows novel methodologies to obtain distance maps on raw fisheye images using a CNN. We will tackle this task by setting up a *structure-from-motion* (*SfM*) framework and use the concept of view synthesis. The entire approach is self-supervised. The problem is more challenging than stereo-based approaches as the network also needs to solve the relative poses between the source images to reconstruct the target. This inherently contains the task of visual odometry, which will be part of our perception stack. At last, in this chapter, we generalize the approach to consider any camera geometry of choice described in Section 2.1.1 for the projection operation involved in the view-synthesis process. With this framework, we extend our work on public datasets such as KITTI [11], and achieve accurate depths, and outperform all previously published self-supervised methods.

In **Chapter** 5, we focus on improving the geometry cues by leveraging semantics guidance. We reason about the L_1 loss function and solve the infinite depth issue, which causes holes during inference, by incorporating the semantic information during view synthesis. We modify the reconstruction loss and replace it with a robust general loss function and obtain significant gains in accuracy. We examine how to leverage more directly the *semantic* context of the scene to guide geometric representation learning while remaining in the self-supervised regime. One of the thesis's main goals is to target a real-time distance estimation convolutional neural network (CNN) design that can be deployed in millions of vehicles having its own set of cameras. Later in this chapter, to do so, we develop a novel camera geometry adaptive multi-scale convolution to incorporate the camera parameters into the self-supervised *SfM* framework. We improve upon the previous work and obtain state-of-the-art results on the KITTI and WoodScape datasets.

In **Chapter** 6, we look into the localization aspect of an autonomous car. We focus on the 2D object detection task in fisheye cameras and perform the first detailed study on object detection on fisheye cameras for AD scenarios. We present novel representations of bounding boxes on raw fisheye camera images *i.e.*, oriented bounding box, ellipse, and generic polygon for object detection. To encourage further research in this direction, we will also make a public release of the dataset comprising 10, 000 images with annotations for all the object representations.

In **Chapter** 7, we develop a whole scene understanding MTL framework for surround-view camera systems named *Omnidet*. It would comprise all the six tasks listed in the thesis's goal. The developed CNN model is real-time capable on an automotive embedded platform. The entire perception system is *using cameras only*. It is a distinct approach considering the immense challenge of AD: one which does not rely on infrastructures such as high-definition maps or extremely costly sensor payloads, yet can perform complex driving tasks using cameras only. It will be the first six-task perception network on fisheye cameras evaluated on the WoodScape dataset. We transfer the framework to a five-task network on the public datasets KITTI and CityScapes and establish state-of-the-art results on the KITTI Eigen depth benchmark and the KITTI odometry benchmark. We apply white and black box attacks on the MTL framework to further analyze the system's robustness on adversarial attacks, considering both targeted and untargeted scenarios.

In **Chapter** 8 we will identify the limitations of the developed system and discuss the challenges and shortcomings of the approach. Finally, in **Chapter** 9 we will draw up a conclusion based on the discussion brought in the preceding chapters.

Chapter 2

Background

Contents

This thesis aims to build a perception system that constitutes a **Level 3** AD stack using a multi-task CNN model, covering the necessary modules for near-field sensing use cases like parking or traffic jam assistance. We propose to design a multi-task model of six primary tasks necessary for an autonomous driving system: depth/distance estimation, visual odometry, semantic segmentation, motion segmentation, object detection, and lens soiling detection. This chapter will provide in-depth basic intuitions of the tasks and the terminologies used in the further chapters of the thesis. The latter part will focus on the datasets and the evaluation metrics. This specific background will determine our strategy on which quality measure is more meaningful regarding safety and path planning for automated parking.

Perception, localization and mapping, sensor fusion, path planning, and decision control are vital components of automated driving. Various perception tasks are required to provide a robust system covering a wide variety of scenarios. In general, a perception system comprises geometric and semantic scene understanding of the environment. Cameras are a dominant sensor for perception since roadway infrastructure is traditionally designed for human visual perception. Semantic tasks such as object detection [36, 37, 38, 39] (detecting pedestrians, vehicles, and cyclists, etc. with bounding boxes), semantic segmentation [40, 41, 42] (pixel-wise labeling of road, lanes, and curbs, etc.) and soiling segmentation [43] (pixel-wise labeling of opaque and transparent) usually falls under the semantic branch, wherein fisheye cameras mounted low on a vehicle are susceptible to lens's soiling due to the splash of mud or water from the street. These are some of the significant tasks that help build a visual perception system.

The semantic tasks typically require a large annotated dataset covering various objects. However, it is practically infeasible to cover every possible object. Thus, generic object detection using geometric cues like motion or depth for rare objects is added to the perception system. Due to the multi-task training, these tasks will not only complement the detection of standard objects but also provide the final model with higher robustness.

Motion is a dominant cue in automotive scenes, and it requires at least two frames or the use of dense optical flow [44, 45, 46, 47, 48, 49] (detect motion and estimate velocities of moving objects). In the case of geometric task *i.e.* depth estimation [50, 51, 52, 53, 54, 55] (distance in a real-world from ego vehicle). Finally, the visual odometry task is required to place the detected objects in a temporally consistent map.

In general, all the tasks mentioned earlier are mainly demonstrated on pinhole images in current research approaches. There is minimal work in the area of fisheye perception. In the following sections, we look into these tasks' background and ground principles and the difficulties in applying these tasks onto the fisheye cameras.

2.1 Fisheye Camera and Geometry

The development of fisheye cameras has a long history. Wood initially coined the term fisheye in 1908 and constructed a simple fisheye camera [56], a fact that is acknowledged in the naming of the recently released *WoodScape* dataset of automotive fisheye camera videos [12]. This water-based lens was replaced with a hemispherical lens by Bond [57], and thus began the optical development of fisheye cameras. Miyamoto [58] provided early insight into the modeling of geometric distortion in fisheye cameras, suggesting the use of equidistant, stereographic, and equisolid models. These models were already known in the field of cartography (*e.g.* [59] and many others).

Rise of Fisheye Cameras: There has been a significant rise in the usage of fisheye cameras in various automotive applications [60, 61, 62], surveillance [63] and useful applications in robotics [64], including robotic localization [65], simultaneous localization and mapping [29, 66, 67, 68], visual odometry [69], due to their large field-of-view (FoV). Recently, several computer vision tasks on fisheye cameras have been explored including object detection [70, 71], semantic segmentation [72], soiling detection [43], motion estimation [21], image restoration [73], underwater robotics [74], aerial robotics [75] and many other related fields. Depth estimation is an essential task in autonomous driving as it is used to avoid obstacles and plan trajectories. While depth estimation has been substantially studied for narrow FoV cameras, it has barely been explored for fisheye cameras [10, 76]. Fisheye cameras offer a significantly wider FoV than standard cameras, often with 180° FoV or even greater. This can offer several advantages, in particular, that fewer cameras can be used to achieve complete surround-view coverage.

Need for Application of Perception Tasks on Fisheye Cameras

Surround-view fisheye cameras have been deployed in premium cars for over ten years, starting from visualization application on dashboard display units to provide near-field perception for automated parking. Fisheye cameras have a strong radial distortion that cannot be corrected without disadvantages, including reduced FoV and resampling distortion artifacts at the periphery [3]. Appearance variations of objects are larger due to the spatially variant distortion, particularly for close-by objects. Thus fisheye perception is a challenging task, and despite its prevalence, it is comparably less explored than pinhole cameras.

Different cameras with different FoV's are used to handle a wide variety of automotive use cases. The most common ones are around 100° hFoV cameras used for front camera sensing and 190° hFoV fisheye lens cameras for surround-view sensing. Due to their moderate to large FoV, these cameras suffer from lens distortion (see Figure 2.1), whose main component is typically radial distortion and minor tangential distortion.

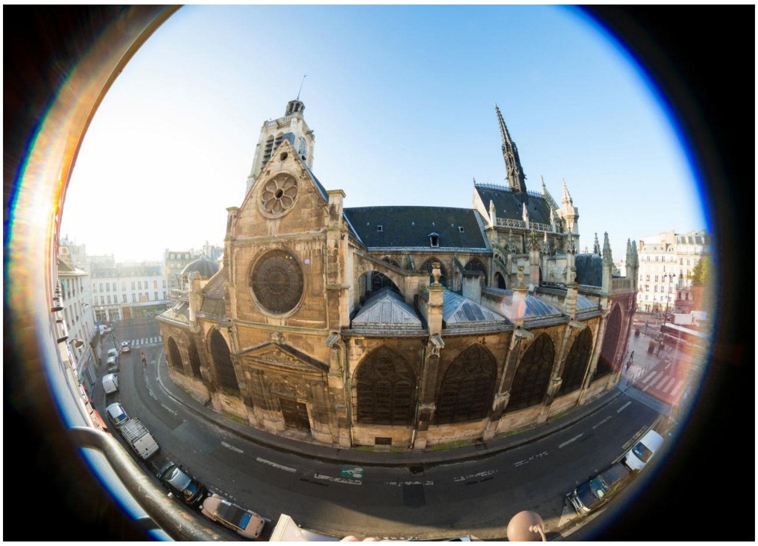


Figure 2.1 **An overview of an old church in Paris captured using a fisheye lens. Figure reproduced from [77].**

2.1.1 Fisheye Lens Projection Models

Several models have been developed to describe fisheye lenses. We can consider them in several classes. For example, we could consider a class of *on-image* models, in which the fisheye projection is measured as a deviation from pinhole projection, *e.g*. [78, 79]. Alternatively, we could consider a model in which the ray projection angle is manipulated at the projection center (*e.g*. [12, 80]). We may refer to these models as *ray deviation* models. Others still propose the use of a series of projections onto different surfaces to model fisheye distortion, for example [81, 82, 83], which we can refer to as *geometric models*.

In the case of cameras with a more standard FoV, there is a very common geometry associated with them, the *pinhole* model. One may first consider the intersection of a ray with a single planar surface at some fixed distance from the projection center. All models of the distortion due to the lens for such cameras are designed to shift the intersection point radially from the projection center on the plane (if one ignores tangential distortion). There are some varying proposals on what a model for this may look like, as discussed in the likes of [84]. However, the geometry for such standard FoV cameras is fixed as a ray-plane intersection. The role of geometric distortion correction is to find the deviation between the real camera and the hypothesized pinhole camera.

In a way, fisheye camera development in vision has been complicated by the lack of a single unifying geometry. There is a large set of models with different properties to describe fisheye projection and its radial distortion. This section aims at explaining some of these models, group them logically, and even highlight a couple of the more recently proposed models. They are almost re-derivations of existing models or describe projections that have already been known in different fields of science.

Notation and Terminology

Matrices are denoted by $A \in \mathbb{R}^{m \times n}$. The usual notation for ordinary vectors $\mathbf{v} \in \mathbb{R}^n$ will be used, represented as n-tuples. Specifically, points in $\mathbb{R}^3$ will be denoted as

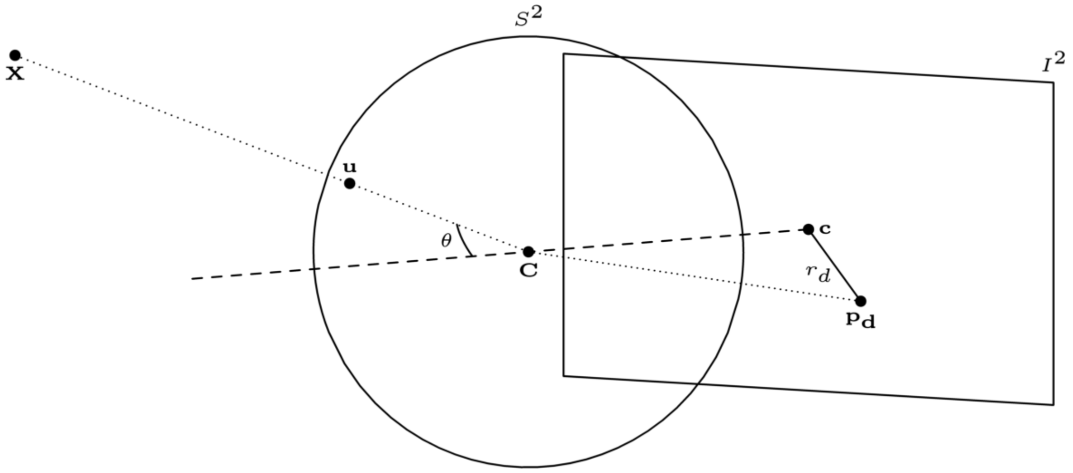


Figure 2.2 **Relationship between fisheye image point and point on the unit sphere. p** and **u** are equivalent points on the fisheye image and unit sphere, respectively, with **u** laying on the same ray as **X**

.

$\mathbf{X} = (X, Y, Z)^\mathsf{T}$. We will use the same notation for points/vectors in $\mathbb{P}^n$, where necessary making it clear in the text that they should be interpreted as $(n+1)$-homogeneous vectors. Unit vectors are represented by the usual carat $\hat{\mathbf{v}} = (\hat{v}_1, \hat{v}_2, \hat{v}_3)^\mathsf{T}$.

We will rely heavily on points defined on the unit sphere embedded in Euclidean 3-space (itself embedded in $\mathbb{R}^3$). More formally, the unit sphere is $S^2 = \{\mathbf{u} \in \mathbb{R}^3 \mid \|\mathbf{u}\| = 1\}$, and thus represented as a 3-vector $\mathbf{u} = (u_1, u_2, u_3)^\mathsf{T}$ of unit length. However, for clarity we forego the unit vector notation as it is implicit for points on the unit sphere.

In fisheye cameras, the lens views with FoVs >180°, and observed rays cannot all pass through a single flat image plane. Therefore, we cannot consider points on a projective plane, as they cannot encompass the entire FoV of a fisheye camera. We can define a mapping from defining a mapping from $\mathbb{R}^3$ to the fisheye image as

$$\pi : \mathbb{R}^3 \to I^2$$

A true inverse is naturally not possible. However, we can define an unprojection mapping from the image domain to the unit central projective sphere

$$\pi^{-1} : I^2 \to S^2$$

Here, $I^2 \subset \mathbb{R}^2$ and $S^2 \subset \mathbb{R}^3$. Figure 2.2 demonstrates the relationship between the image plane and the unit sphere.

For the pinhole and the fisheye models that will follow, to simplify the math, note that we do not consider a separate f parameter for each image direction (*i.e.*, f_x and f_y, we assume that the pixel aspect ratio is unit). However, it is easy to adapt the fisheye equations to include a non-unit aspect ratio, but that would not serve the purpose of this thesis. In addition, we do not consider the distortion center $\mathbf{c} = (c_x, c_y)^\mathsf{T}$, as this is just a translation on the image plane and does not affect the model performance.

2.1.2 Classical Geometric Models

We refer to the models discussed in this section as *classical*, as they have been researched for at least six decades [58]. One could also include the equisolid-angle model. However, we do not employ it for the scope of this work, and the readers are instead referred to [85, 86].

Pinhole Camera Model

For lenses with a moderate FoV ($< 120°$), the Brown–Conrady model [87] is commonly used as it models both radial and tangential distortion. For larger FoV, this distortion model typically breaks down or requires very high polynomial orders. The KITTI dataset's calibration uses this model based on OpenCV's [88] implementation. Assuming a point $\mathbf{X} = (X, Y, Z)^\mathsf{T}$ in the camera coordinate system, the pinhole model is

$$\mathbf{p} = \left(\frac{fX}{Z}, \frac{fY}{Z}\right)^\mathsf{T} \tag{2.1}$$

or, if we consider it as a radial function

$$r_u = f_p \tan\theta \tag{2.2}$$

where θ is the field angle of the projected ray, note that the parameter f_p is sometimes referred to as the focal length. However, it has little to do with the optical focal length of the physical lens system (which can often be made up of many lens elements). f_p is a scaling factor that converts from a projection surface at a unit distance from the project center to the pixel coordinates of the camera.

In this model, the projection function $X_c \mapsto \Pi(X_c) = p$ maps a 3D point $\mathbf{X} = (X, Y, Z)^\mathsf{T}$ in the camera coordinate system to a pixel $p = (i, j)^T$ in the image coordinates. It is calculated in the following way:

$$\begin{aligned} x = X/Z, \quad y &= Y/Z \\ x' = x(1 + k_1r^2 + k_2r^4 + k_3r^6) &+ 2p_1xy + p_2(r^2 + 2x^2) \\ y' = y(1 + k_1r^2 + k_2r^4 + k_3r^6) &+ p_1(r^2 + 2y^2) + 2p_2xy \\ i = f_x \cdot x' + c_x, \quad j &= f_y \cdot y' + c_y \end{aligned}$$

where k_1, k_2, and k_3 are radial distortion coefficients, p_1 and p_2 are tangential distortion coefficients of the lens, $r^2 = x^2 + y^2$, f_x, f_y are the focal lengths and c_x, c_y are the coordinates of the principal point.

Equidistant Projection

In the *equidistant fisheye model*, the projected radius r_d is related to the field angle θ through the simple scaling by the equidistant parameter f_e (see Figure 2.3). *i.e.*

$$r_d = f_e\theta \tag{2.3}$$

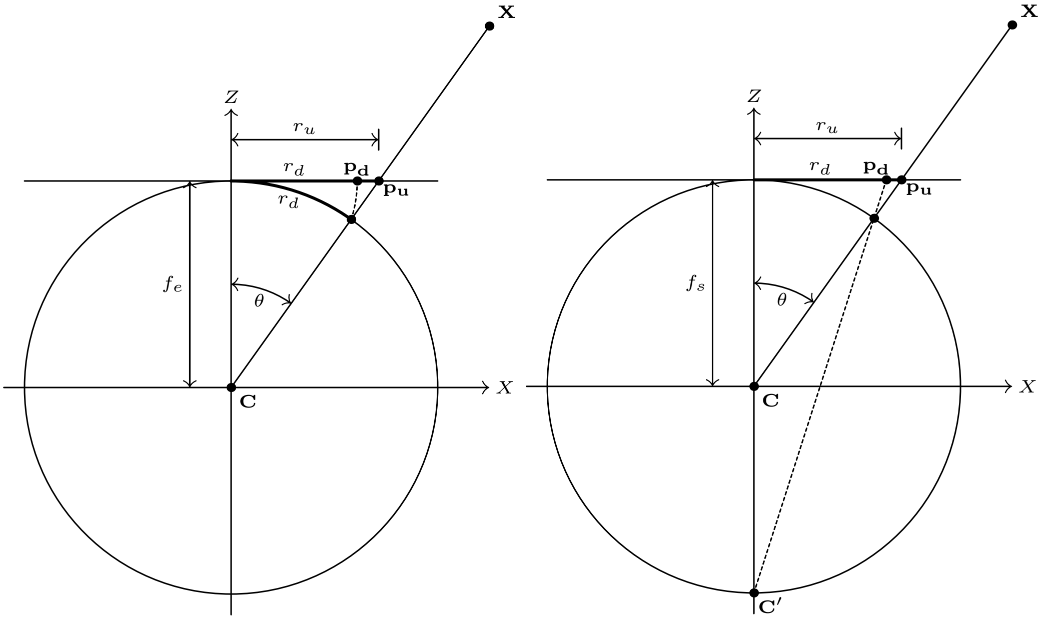


Figure 2.3 **Equidistant and Stereographic fisheye projection models.**

and thus

$$\begin{aligned} \pi(\mathbf{X}) &= \frac{f_e\theta}{d}\begin{bmatrix} X \\ Y \end{bmatrix} \\ d &= \sqrt{X^2+Y^2} \\ \theta &= \operatorname{acos}\left(\frac{Z}{\sqrt{X^2+Y^2+Z^2}}\right) \end{aligned} \tag{2.4}$$

Stereographic Projection

As with the equidistant model, in *stereographic projection*, the center of the projection of **X** to the projection sphere is **C** (Figure 2.3). Consider that the image plane has a tangential point along the Z-axis (optical axis). There is a second central projection to the image plane in stereographic, with the antipodal point of the tangential point forming the center of projection. It is essentially a pinhole projection with a focal length of $2f_s$. The stereographic projection is therefore described by

$$r_d = 2f_s \tan\left(\frac{\theta}{2}\right) \tag{2.5}$$

$$\begin{aligned}\pi(\mathbf{X}) &= \frac{r_d}{\sqrt{X^2+Y^2}}\begin{bmatrix}X\\Y\end{bmatrix}\\ &= \frac{2f_s\tan\frac{\theta}{2}}{\sqrt{X^2+Y^2}}\begin{bmatrix}X\\Y\end{bmatrix}\\ &= \frac{2f_s\tan\left(\frac{1}{2}\text{atan}\left(\frac{\sqrt{X^2+Y^2}}{Z}\right)\right)}{\sqrt{X^2+Y^2}}\begin{bmatrix}X\\Y\end{bmatrix}\\ &= \frac{2f_s\frac{\sqrt{X^2+Y^2}}{\sqrt{X^2+Y^2+Z^2}+Z}}{\sqrt{X^2+Y^2}}\begin{bmatrix}X\\Y\end{bmatrix}\end{aligned} \tag{2.6}$$

$$\pi(\mathbf{X}) = \frac{2f_s}{Z+||\mathbf{X}||}\begin{bmatrix}X\\Y\end{bmatrix} \tag{2.7}$$

where d and θ are equally defined as in (2.4). The inverse of (2.5), which we shall need later, is derived as

$$\theta = 2\text{atan}\left(\frac{r_d}{2f_s}\right) \tag{2.8}$$

Orthographic Projection

Similar to the previous projections models, the *orthographic projection* begins with a projection to the sphere (Figure 2.4). An orthogonal projection to the plane follows this. The orthographic projection is therefore described by

$$r_d = f_o\sin\theta \tag{2.9}$$

and thus

$$\pi(\mathbf{X}) = \frac{f_o}{||\mathbf{X}||}\begin{bmatrix}X\\Y\end{bmatrix} \tag{2.10}$$

Extended Orthographic Model

The *Extended Orthographic Model* [89], as demonstrated by Figure 2.4, extends the classical orthographic model by freeing the projection plane from being tangential to the projection sphere, allowing an offset λ. The distorted projection remains the same as equations (2.9) and (2.10). However, the relationship between the distorted and undistorted radial distances is given by

$$r_d = \frac{f_o}{\sqrt{(\lambda+f_o)^2+r_u^2}} \tag{2.11}$$

and thus is an on-image mapping given by

$$\begin{bmatrix}x_d\\y_d\end{bmatrix} = \frac{f_o}{r_u\sqrt{(\lambda+f_o)^2+r_u^2}}\begin{bmatrix}x_u\\y_u\end{bmatrix} \tag{2.12}$$

2.1.3 Algebraic Models

We provide a short discussion on algebraic models of fisheye cameras, specifically polynomial models and the division model. We provide the polynomial model

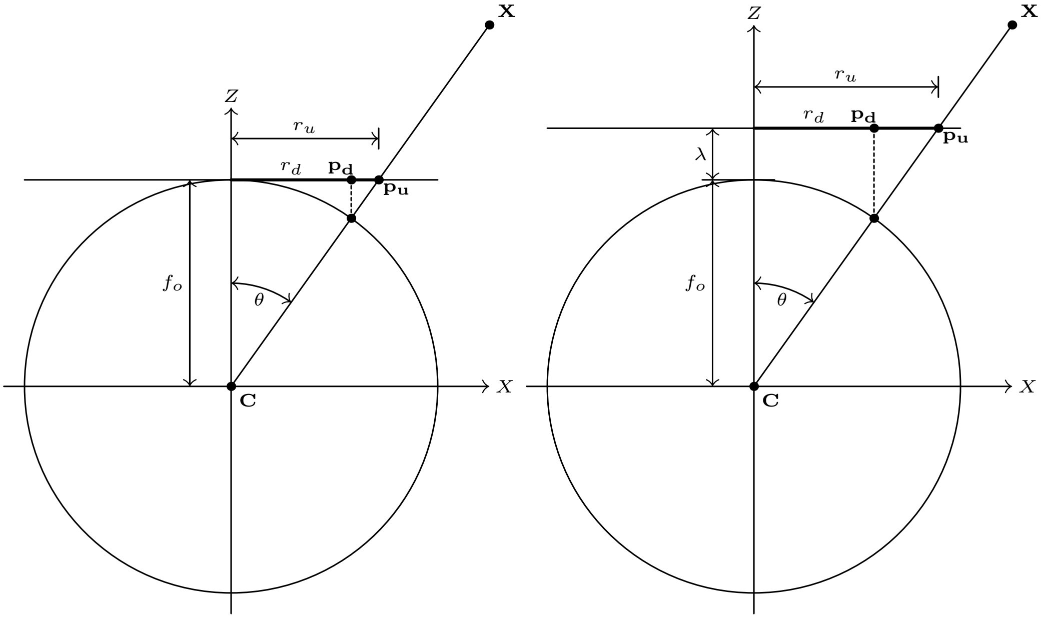


Figure 2.4 **Othographic and Extended Othographic fisheye projection models.**

discussion for completeness, though we concentrate on the geometric models for the remainder of the chapter.

Polynomial Models

The classical *Brown–Conrady model* of distortion for non-fisheye cameras [87, 90] uses an odd-termed polynomial, $r_d = P_n(r_u)$, to describe the radial distortion on the image (*i.e.* mapping r_u to r_d), where P_n represents some arbitrary n-th order polynomial. Despite its age, the Brown-Conrady model is the standard distortion model in software implementations for non-fisheye cameras [88, 91]. To account for fisheye distortion, an on-image polynomial model known as the *Polynomial Fisheye Transform* (PFET) was proposed in [78]. The difference between the PFET and the Brown-Conrady model is that the PFET allows both odd and even exponents to account for the added distortion encountered in fisheye cameras.

A class of polynomial fisheye models exists, in which the mapping of the field angle to the image plane is via a polynomial, *i.e.* $r_d = P_n(\theta)$. For example, Kannala-Brandt [80] (and as implemented in the popular *OpenCV* software [88]) propose an polynomial model of order $n = 5$, or more, with odd exponents only. In [12], an $n = 4$ polynomial containing both even and odd exponents is proposed. Neither model used a constant coefficient term in the polynomial. In [92] a fifth-order polynomial is proposed, but they reduce it to four independent parameters if the fisheye radius and the FoV are known. All of the above could be interpreted as a generalization of the equidistant model, which is a first-order polynomial. In this case, the projection sphere is replaced by some surface defined by the given polynomial. However, this is forcing a geometric interpretation with little utility.

Division Model

The *division model* [84] of radial distortion gained some popularity due to the excellent property that, at least for the single parameter variant, straight lines project to circles in the image [93, 94, 95]. For many lenses, the single parameter variant performs very well [96]. It is given by

$$r_u = \frac{r_d}{1 - ar_d^2} \tag{2.13}$$

$$r_d = \frac{r_u}{1 + ar_u^2} \tag{2.14}$$

where a is the division model parameter. This was extended in [97] by adding an additional scaling parameter, which improved the modeling performance for certain types of the fisheye lenses. Note that the division model was presented as an *on-image* mapping, though it can be expressed as the projection function

$$\begin{aligned} \pi(\mathbf{X}) &= \begin{bmatrix} \frac{fr'_d X}{r'_u} \\ \frac{fr'_d Y}{r'_u} \end{bmatrix} \\ r'_u &= \sqrt{X^2 + Y^2} \\ r'_d &= \frac{r'_u}{1 + ar'_u} \end{aligned} \tag{2.15}$$

2.1.4 Geometric Models

A set of more recent (at least, from the last couple of decades) fisheye camera models is also considered. Further reading on some of these models can be found, *e.g.*, in [83], which also gives valuable information on the set of valid points for each projection.

Field-of-View Model

The FoV model [79] is defined by

$$r_u = \frac{\tan(r_d\omega)}{2\tan\frac{\omega}{2}} \tag{2.16}$$

where r_u is the distance of the undistorted image point (pinhole image point) to the distortion center, r_d is the corresponding distorted image point distance, and ω is the model parameter. The parameter ω approximates the camera field of view, though not exactly [79].

This is an *on-image* model, like the Division Model, where r_u and r_d define undistorted and distorted radii on the image plane. Alternatively, it can be expressed as a projection function [83]

$$\begin{aligned} \pi(\mathbf{X}) &= \begin{bmatrix} \frac{fr'_d X}{r'_u} \\ \frac{fr'_d Y}{r'_u} \end{bmatrix} \\ r'_u &= \sqrt{X^2 + Y^2} \\ r'_d &= \frac{\operatorname{atan2}(2r'_u \tan(\omega/2), z)}{\omega} \end{aligned} \tag{2.17}$$

Note that r'_u and r'_d are related to the undistorted r_u and distorted r_d image plane radial distances, but are not quite the same, due to the scaling by f.

Unified Camera Model

The UCM was initially used to model catadioptric cameras [81]. Later it was shown to be helpful when modeling fisheye cameras [98, 99]. It has been shown to perform well across a range of lenses [100]. The geometry of the projection is two-step. First, the point **X** is projected to a unit sphere, followed by a projection to a modeled pinhole camera. The UCM projection is given by

$$\pi(\mathbf{X}) = \frac{\gamma}{Z + \xi||\mathbf{X}||} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{2.18}$$

though [83] propose a more numerically stable formulation. ξ is the distance from the center of the unit sphere to the center of the pinhole projection, and γ is the focal length of the secondary pinhole projection. If $\xi = 0$, this model degrades to the pinhole model.

Enhanced Unified Camera Model

The UCM was extended by the Enhanced UCM [82], which replaced the spherical projection with a projection to an ellipsoid (or, in fact, a general quadratic surface), and was able to demonstrate some accuracy gain. The E-UCM is given by [83]

$$\pi(\mathbf{X}) = \frac{f}{\alpha d + (1-\alpha) Z} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{2.19}$$

where $d = \sqrt{\beta(x^2 + y^2) + z^2}$.

Double-Sphere Model

Later still, the UCM was extended again by the double-sphere (DS) model [83], which added a second spherical projection to enable more complex modeling

$$\begin{aligned} \pi(\mathbf{X}) &= \frac{f}{\alpha d_2 + (1-\alpha)(\xi d_1 + Z)} \begin{bmatrix} X \\ Y \end{bmatrix} \\ d_1 &= \sqrt{x^2 + y^2 + z^2} \\ d_2 &= \sqrt{x^2 + y^2 + (\xi d_1 + Z)^2} \end{aligned} \tag{2.20}$$

Convincing results are presented in [83] to demonstrate the effectiveness of the double-sphere model.

Summary of Radial Distortion Models

The radial distortion models are summarized below:

- Polynomial: $r(\theta) = a_1\theta + a_2\theta^2 + a_3\theta^3 + a_4\theta^4$
- UCM: $r(\theta) = f \cdot \sin\theta / (\cos\theta + \xi)$
- eUCM: $r(\theta) = f \cdot \frac{\sin\theta}{\cos\theta + \alpha\left(\sqrt{\beta \cdot \sin^2\theta + \cos^2\theta} - \cos\theta\right)}$
- Rectilinear: $r(\theta) = f \cdot \tan\theta$
- Stereographic: $r(\theta) = 2f \cdot \tan(\theta/2)$

- Double Sphere: $r(\theta) = f \cdot \frac{\sin\theta}{\alpha\sqrt{\sin^2\theta + (\xi + \cos\theta)^2} + (1-\alpha)(\xi + \cos\theta)}$

2.1.5 Other Models

While we have discussed many of the more popular fisheye projection models, the list is still incomplete. We have omitted some models are only rarely used. For example, Bakstein and Pajdla [101] proposed two extensions to the classical models. Firstly, they allowed a second parameter in the stereographic model. Then by trying various combinations of classical models for a given fisheye camera, they propose what is essentially a weighted average of the stereographic and the equisolid angle projection. A logarithm-based *Fisheye Transform* (FET) was also proposed in [78], though the accuracy was low compared to other models. The hyperbolic sin-based model proposed in [102], and later used for wide-angle cameras [103], is not discussed here. The cascaded one-parameter division model [104] is also not mentioned.

2.1.6 Similarity between Models

With the proliferation of fisheye models, it is natural to wonder if there is a commonality between some of the models or even if there has been repetition in the development of the models.

General Perspective Projection and Fisheye Models

The unified camera model is in a class of general vertical perspective projections of a sphere, which is known in the fields of geodesy and cartography [105, 106], with the addition of the trivial step of central projection to the spherical surface. The stereographic and the orthographic projections belong to this class as well. The stereographic projection has the pinhole projection center on the sphere's surface, while the orthographic projection has an infinite focal length (hence the term orthographic). The link between the stereographic projection and the UCM is described in [81].

Let us begin by examining the general vertical perspective projection, described by Figure 2.5. The pinhole camera is offset along the Z-axis by a distance of d. The projection to the sphere is given by

$$\mathbf{u}' = f_s \frac{\mathbf{X}}{||\mathbf{X}||} \tag{2.21}$$

Here we use $\mathbf{u}' = (u'_x, u'_y, u'_z)^\intercal$ for the point on the sphere of radius f_s, to distinguish it from $\mathbf{u}$ used previously to denote a point on the unit sphere. The point $\mathbf{p_d}$ is the pinhole projection of $\mathbf{u}'$

$$\pi(\mathbf{X}) = \frac{f_p}{u'_z + d}\begin{bmatrix} u'_x \\ u'_y \end{bmatrix} = \frac{f_p}{Z + \frac{d}{f_s}||\mathbf{X}||}\begin{bmatrix} X \\ Y \end{bmatrix} \tag{2.22}$$

The $+d$ translates the point $\mathbf{u}'$ from the sphere to the pinhole coordinate system. Thus, with the two parameters $\gamma = f_p$ and the $\xi = d/f_s$, we have (2.18), the UCM. Additionally, if we constrain the pinhole camera plane to be on the surface of the sphere (i.e. $d = f_s$), and make $f_p = 2f_s$, we get the stereographic equation (2.7).

The Enhanced UCM [82] extended the UCM by projecting to an ellipsoid instead of a sphere. Again, this type of projection is known in geodesy and cartography for a long time [105, 106] as *ellipsoidal* general perspective projections. We will not re-derive the

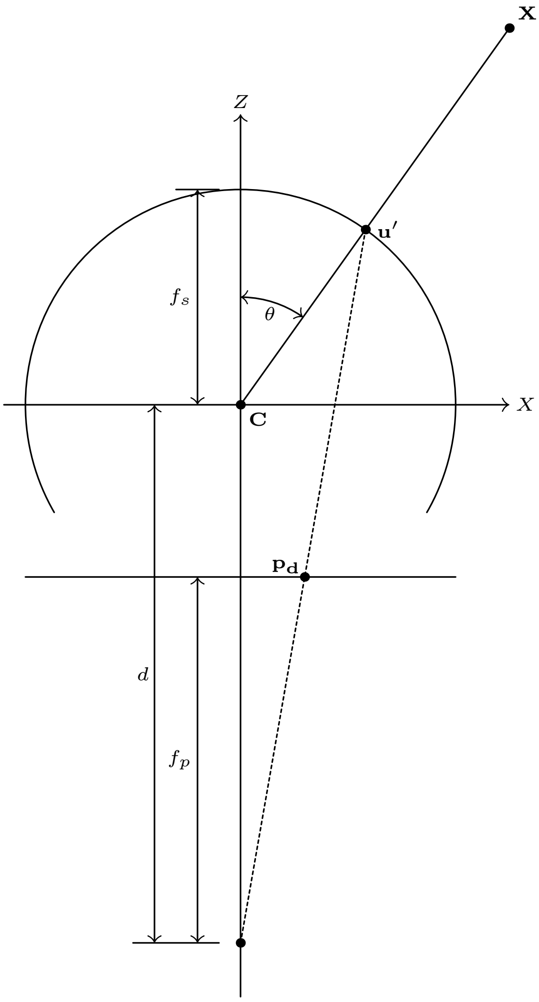


Figure 2.5 **The general perspective mapping.**

equations here but would refer the reader to the source material. As mentioned, the DS model [83] extends the UCM by adding a second projection sphere to model more complex optics.

Thus the UCM, the E-UCM, and the DS models of fisheye lenses can be considered as generalizations of the stereographic camera model. It may be even more correct to say that they all (UCM, E-UCM, DS, division, and stereographic models) are part of a class of general perspective models. If we allow f_s to approach infinity, then (2.22) becomes the pinhole projection model. If we allow f_p (and thus also d) to go to infinity, then we get the orthographic projection.

Figure 2.6 graphically shows the relationships between the various fisheye models and the general perspective projection. The Division Model stands out, as it was not initially designed as a geometric projection model, though its equivalence to the Stereographic Model is demonstrated in the next section. It should also be noted that the E-UCM is not restricted to ellipsoids but, depending on the parameters, may also be represented by hyperboloid or paraboloid projection surfaces. However, this is such a minor differentiation that we consider this still to be equivalent to the Ellipsoidal General Perspective Projection.

In Figure 2.6, we have attempted to provide a map of geometric fisheye models that

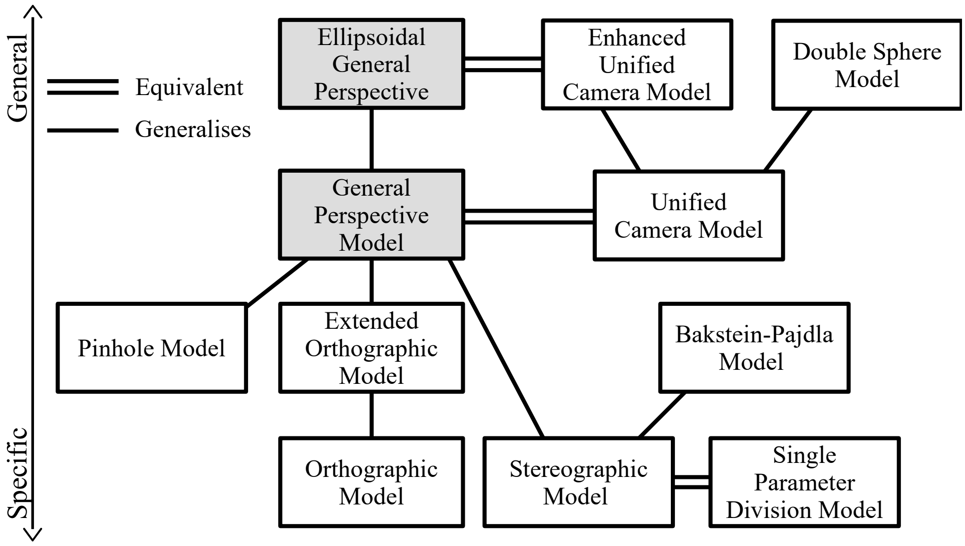


Figure 2.6 **The relationship between the various fisheye models and the general perspective projections.** Double line indicates that two models are equivalent, and single line indicates a generalization/specialization.

are related to the General Perspective Projection. For a developer, this could be seen as a tool to guide the choice of model for a given application. One could attempt to use the simpler, more specialized models and, depending on the specific application, extend the development to one of the more general models in the case that errors remain high for a given camera model following calibration.

Stereographic and Division Models

We can combine the pinhole projection (2.2) with the inverse of the stereographic model (2.8) to give

$$r_u(r_d) = f_p \tan\left(2\text{atan}\left(\frac{r_d}{2f_s}\right)\right) \tag{2.23}$$

Elementary trigonometry yields

$$r_u = \frac{f_s}{f_p}\frac{r_d}{1 - \frac{r_d^2}{4f_p^2}} \tag{2.24}$$

Let us compare this with the single parameter division model. If we allow $a = 1/4f_p^2$, this is the same as the division model, (2.13), up to a scaling factor f_s/f_p, which is discussed in [86].

Equidistant and Field-of-View Models

Consider the radial pinhole projection given by (2.2), and the equidistant fisheye projection model (2.3). Combining the two to a similar form as the FoV model (2.16)

$$r_u = f_p \tan\frac{r_d}{f_e} \tag{2.25}$$

As f_p and f_e are free parameters, determined through calibration, we can set them to

$$f_p = \frac{1}{2\tan\frac{\omega}{2}} \quad \text{and} \quad f_e = \frac{1}{\omega} \tag{2.26}$$

Thus we see that (2.16) and (2.25) are equivalent mapping functions. *i.e.*, the equidistant and the field-of-view model are fundamentally the same model.

Mapping of 3D to 2D for Fisheye Lenses

For fisheye lenses, the mapping of 3D points to pixels universally requires a radial component $r(\theta)$ [107]. The projection is a complex multi-stage process compared to regular lenses and thus we list the detailed steps:

1. The point X_c in camera coordinates is mapped to a unit vector as $S = (s_x, s_y, s_z)^T = X_c / \|X_c\|$.
2. The incident angle against the optical axis (coincident with the Z-axis) $\theta = \frac{\pi}{2} - \arcsin(s_z)$ is computed.
3. The radial function $r(\theta)$ to get the radius on the image plane (typically in pixels) is computed.
4. Given the pixel distortion centre (c_x, c_y), the pixel location is given by $i = r \cdot s_x / \rho + c_x$ and $j = r \cdot s_y / \rho + c_y$ with $\rho = \sqrt{s_x^2 + s_y^2}$.
5. (optional) Depending on the model used in Step 3, an additional distortion correction may needs to be applied.

There is a great number of potential models for application with fisheye cameras. We have mentioned *twenty* models, though this is not yet exhaustive. We have shown that there exists a strong relationship among many of the geometric models—at least seven of the models being related to or directly equivalent to the General Perspective Projection. In addition, we have shown that some of the more recently developed fisheye camera models are mathematically equivalent to the classical fisheye projection functions, being the stereographic and the equidistant models proposed decades ago. One could further theorize the existence of as yet undiscovered fisheye models. For example, from Figure 2.6, you could foresee the unification of the Ellipsoidal General Perspective model and the double sphere by developing a model that consists of sequential projections onto two quadratic surfaces.

The aim of this background is not to diminish the importance of any work in this space. At no point do we make any claim about the accuracy of any of the models. The accuracy of any model depends not just on the model itself, but on the application, on the lens and image sensor types, and the calibration procedure deployed. Instead, we would hope to provide a guide for further development (and perhaps unification) in fisheye projection models. Furthermore, perhaps it can be a valuable source of discussion for a developer considering which model is most appropriate for their camera and application.

2.2 Single-Task Learning (STL)

2.2.1 Depth Estimation

Depth estimation involves estimating the distance to an object (or any plane) at a pixel level as shown in Figure 2.7. For depth estimation, it is very useful to estimate the norm ($\sqrt{x^2 + y^2 + z^2}$) instead of z, because for fisheye images, the z value can be (close to) zero for FoV > 180°, which leads to mathematical problems, because all models have some direct or indirect division by Z. Instead, the norm is always

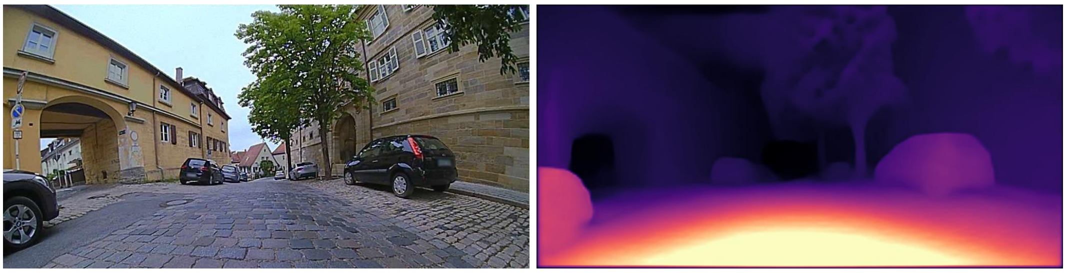


Figure 2.7 **A qualitative sample of distance/depth estimation on WoodScape [12].**

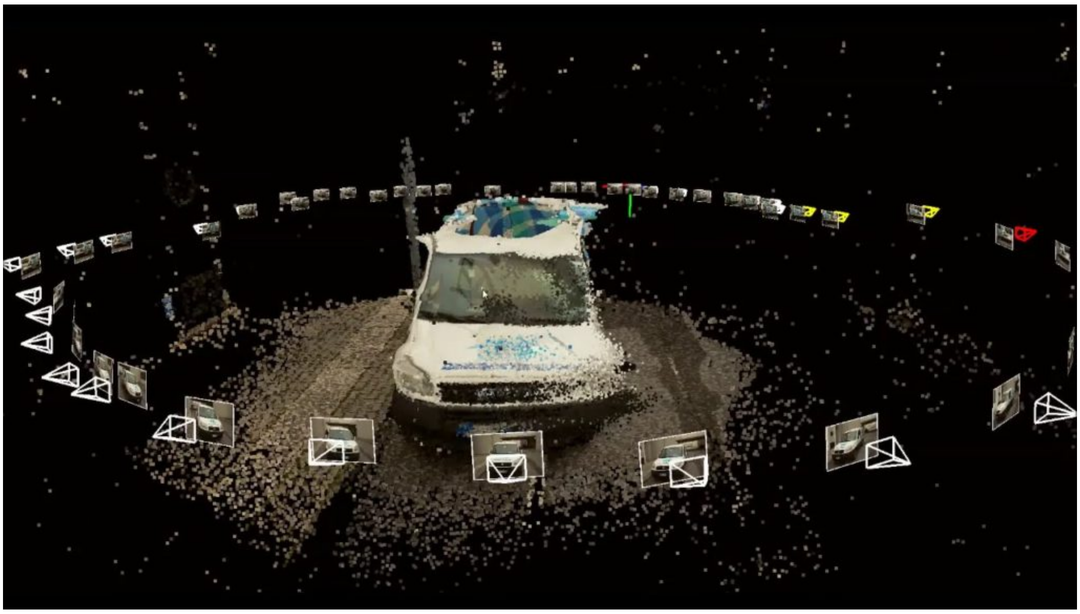


Figure 2.8 **An overview of reconstructed scene in 3D. Figure reproduced from [110].**

zero (except for $x, y, z = 0$) and allows a more numerical stable implementation. Calculating distance relative to a camera plane is still very challenging, but it is critical to unlocking exciting technologies such as autonomous driving, 3D scene reconstruction, and augmented reality. Distance estimation is a crucial requirement in robotics for performing various tasks such as perception, navigation, and planning. Another interesting application would be creating a 3D map that finds its application in Simultaneous Localisation and Mapping [108, 109]; computing depth helps us back-project images from different views into 3D as shown in Figure 2.8. The scene can then be restructured by registering and matching all of the points.

How do we Perceive our World?

Let us begin by discussing how humans view distance in general. Since many of these approaches were derived from our human vision system, this will provide us valuable insights into depth estimation. The formation of an image is similar in both computer and human vision, as shown in Figure 2.9. In theory, as light rays from a source strike objects, they bounce off and move towards the back of our retina, where they are projected and processed in 2D [111], similar to how an image is projected on an image plane. So, how do we calculate distance and comprehend our surroundings in 3D when the predicted scene is in 2D? For example, assume someone is about to punch; we would instinctively know when we are about to be hit and dodge it when

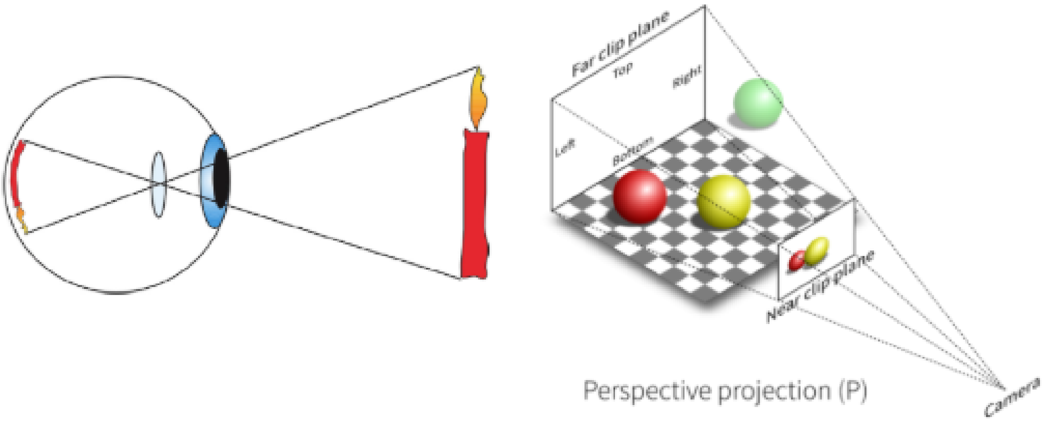


Figure 2.9 **Projecting onto the retina (left). Projecting onto the image plane (right) [112].**

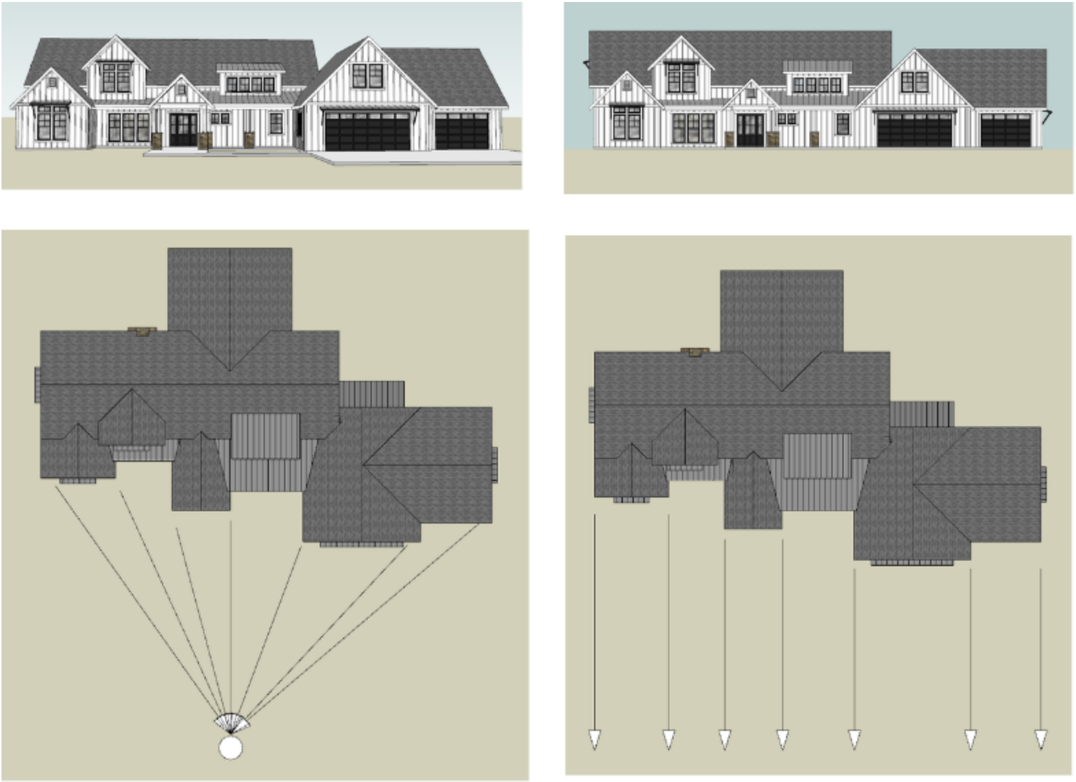


Figure 2.10 **Perspective Projection (Left). Orthographic Projection (Right). Figure reproduced from [113].**

his/her fist gets too close. Alternatively, when driving a car, we might somehow gauge when to step on the accelerator or hit the brakes to maintain a safe distance from so many other drivers and pedestrians [112]. The mechanism at work here is that our brain begins to reason about the incoming visual signals by identifying patterns such as scale, texture, and motion in the scene referred to as *Depth Cues*. There is no distance information in the image, but we can easily interpret and recover depth information. We can tell which parts of the scene are closer to us and which are farther away. Furthermore, these cues allow us to view objects and surfaces that are ostensibly flat on 2D images as 3D [111].

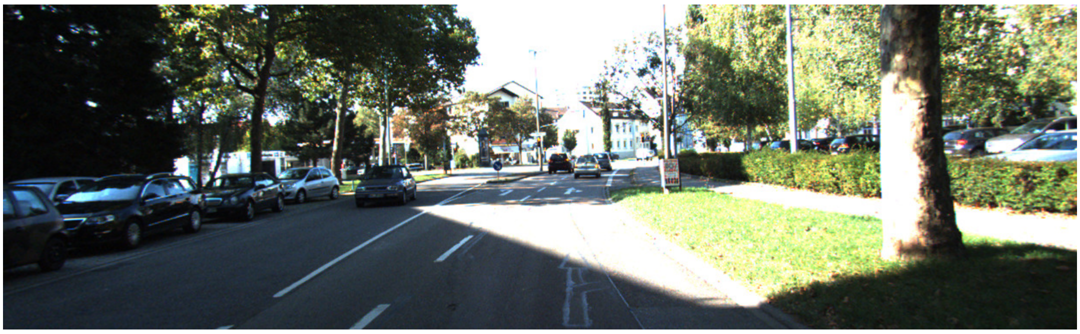


Figure 2.11 **A sample image from KITTI [116] for the illustration of depth cues.**

Monoscopic Depth Cues	*Examples*	*Appear Nearer*	*Appear Farther*
Size of objects	Tree	Larger	Smaller
Texture	Grass patch	High Quality	Low Quality & Blurry
Linear Perspective	Curb	✗	Converge to Horizon

Table 2.1 **Analysis of Monoscopic Depth Cues on a sample from KITTI.**

How to destroy depth from perspectives for human/computer vision?

Depth ambiguity: Understanding depth cues starts with understanding how scenes are projected to perspective view in human and camera vision. An orthographic projection to front view or side view, on the other hand, loses all depth details. Let us consider Figure 2.10, where an observer can tell which side of the house is closer to him/her, as seen in the left image. However, from the right image, it is impossible to discern relative distances. The background may also be on the same plane as the home.

Inferring Depth Using Cues

Basically, there are four types of depth hints: static monocular, motion depth, binocular and physiological hints [114]. We use these signals subconsciously to perceive depth remarkably well. Our perception of depth from a single image mainly depends on the spatial arrangement of a scene. In Table 2.1, we summarize some cues that enable us to understand the distance between various objects. From our every day, it can already feel normal to us. Jonathan *et al*. [115] experimentally illustrates that when the horizon is visible, there is an overwhelming tendency for humans to exploit this cue to perceive depth quickly. By looking at the image in Figure 2.11, we can summarize the following cues depicted in Table 2.1.

Depth Cues from Motion (Motion Parallax)

Motion parallax is a type of depth perception cue in which closer objects appear to move faster than further away objects. It is a type of monocular cue, a depth perception cue that can only be perceived with one eye. This is in contrast to binocular cues, which are depth perception cues that can only be perceived with both eyes open.

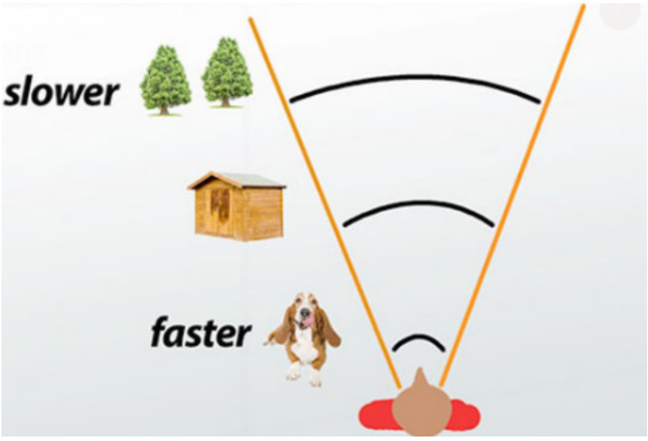


Figure 2.12 **Illustration of motion parallax. Figure reproduced from [117].**

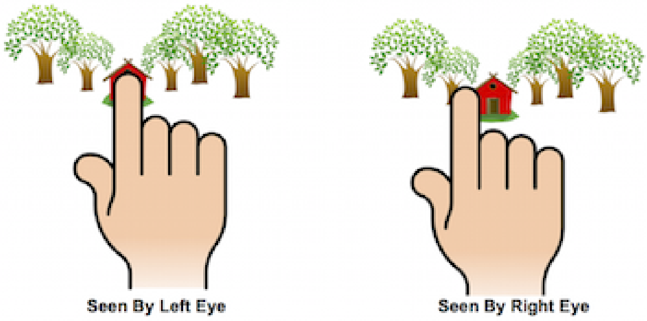


Figure 2.13 **Illustration of Retina disparity [118].**

Motion parallax occurs when objects at different distances from us appear to move at different rates while we are moving (see Figure 2.12). The speed with which an object moves is used to determine its distance. The closer an object is to us, the faster it appears to move. The further an object is from us, the slower it appears to move [117]

Depth Cues from Stereo Vision (Binocular Parallax)

Retina Disparity: Let us look into another fascinating phenomenon that enables us to understand the depth that can intuitively be grasped by demonstrating a simple experiment. If we place our index finger close to our face with one eye closed. Now, repeatedly if we close one and open the other. We can observe that our finger is moving! *Retina disparity* refers to the disparity in vision between our left and right eyes. If we stick out our finger at arm's length and repeat the process, we should note that the shift in finger position becomes less noticeable as shown in 2.13. This experiment provides us with some insight into how stereo vision works. *Stereopsis* is the ability to perceive depth due to two different views of the world. The brain computes distance by comparing images from the retinas of the two eyes. The greater the disparity, the closer things are to us [112].

Depth Estimation in Computer Vision

Depth estimation aims to recover the three-dimensional structure and appearance of objects in imagery by obtaining a representation of the spatial structure of a scene. This is also known as the inverse problem [119], in which we attempt to recover certain unknowns despite the fact that there is insufficient knowledge to define the solution completely, *i.e.*, the mapping between the 2D and 3D views is not unique. We discuss this issue in detail in the following sections. So, how do machines perceive depth? Can any of the ideas discussed above be transferred in some way? Back in the 1990s, the first algorithm with promising results was depth estimation using stereo vision. Since then, Dense stereo correspondence algorithms have made significant progress [120, 121, 122]. The approaches involved using geometry to constrain and reproduce the concept of stereopsis mathematically and in real-time. Scharstein *et al.* [123] summarizes most of these ideas in his survey.

Most research either exploits geometrical cues such as multi-view geometry or epipolar geometry [124] to learn depth. Monocular depth estimation has recently gained popularity due to the use of neural networks to learn a representation that distills depth directly [50]. Accordingly, gradient-based approaches are used to learn depth cues implicitly. Aside from that, there has been significant progress in self-supervised depth estimation [52, 53, 55], which is particularly exciting and revolutionary due to its wide trainability on arbitrary videos. In this approach, we train a model to predict depth by means of optimizing a proxy signal. In the training phase, we do not require any ground truth label and applicability on single images during inference.

2.2.2 Object Detection

Before we get started on creating a cutting-edge model, let us first define object detection. Let us pretend we are building a vehicle detection system for a self-driving car. Assume our car captures an image similar to the one in Figure 2.14. The image from the rear camera essentially conveys that it is a one-way street, and there is a car right next to our rear-end, and quick maneuvers to the right must be prohibited. We should slow down and allow the car to pass through if we need to steer right.

So, what will the car's system do to ensure that this maneuver can occur safely?

It can draw a bounding box around the cars so that the algorithm can determine where the cars are in the image and then decide which direction to proceed to prevent any mishaps.

We aim to perform object detection mainly:

- To detect where the objects are present in the image and locate them.
- To filter out the objects of interest.

Object detection is therefore defined as a model that entails categorizing and localizing various objects in an input image based on their location (see Figure 2.14). It detects the presence of an object in an image and draws a box around it. This typically entails two processes: classifying an object's form and then drawing a box around that object. Object detection tasks are efficiently solved using fully convolutional neural networks. CNN-based bounding box detection can be divided into two categories: single-stage and two-stage approaches. Single-stage methods regress box coordinates and class categories in a single pass. YOLO [38] and SSD [125] are early adopters of single-stage approaches. On the other hand, two-stage networks use explicit loss functions for class-agnostic area proposals accompanied by accurate box coordinates regression. This group includes the R-CNN family of algorithms [126].

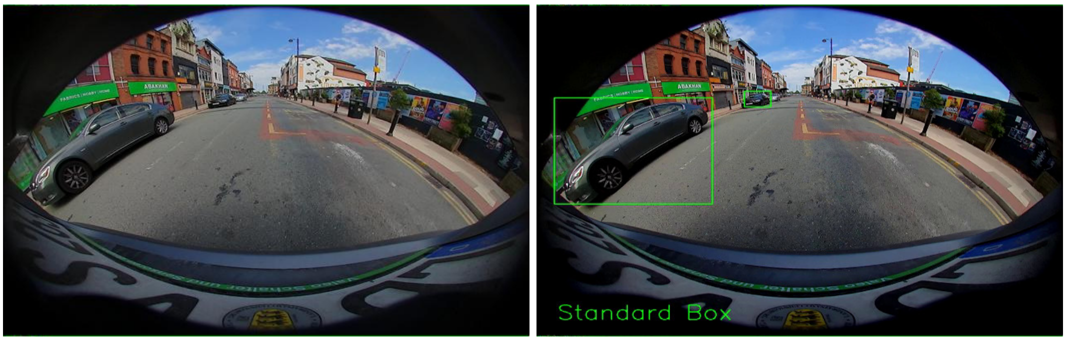


Figure 2.14 **A qualitative input sample (left) and object detection prediction (right) on WoodScape [12].**

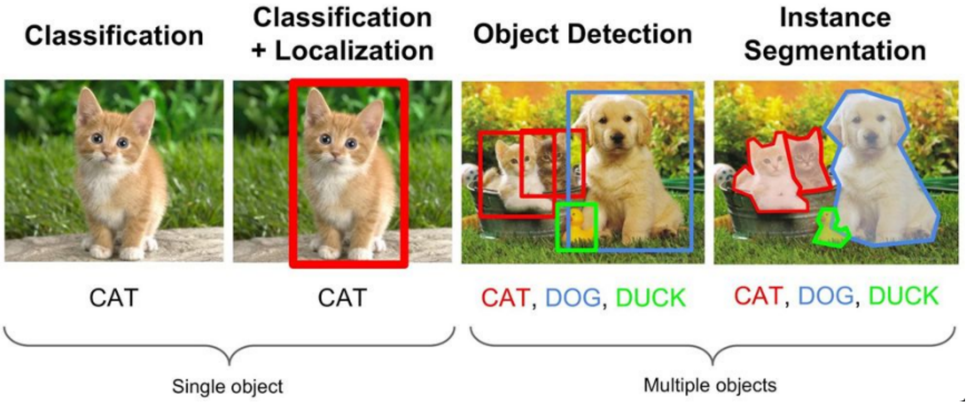


Figure 2.15 **Computer Vision tasks in ascending order of complexity. Figure reproduced from [128].**

2.2.3 Semantic Segmentation

Deep learning has been very effective when working with images as data, and it is now at the point that it outperforms humans in a variety of use-cases. In descending order of complexity, the fundamental problems humans have been involved in solving with computer vision are image classification, object detection, and segmentation, as shown in Figure 2.15. In the task of image classification, we are simply interested in obtaining the labels of all the objects present in an image. Object detection, as discussed in Section 2.2.2 takes it a step further by attempting to detect all objects that are present in an image and the position at which the objects are present using bounding boxes. Image segmentation is even more complex as a task, thus attempting to determine the exact boundary of the objects in the image [127].

Semantic segmentation is the process of assigning a class label to each pixel in an image. These labels could refer to a person, road, curb, pole, and so on, as shown in Figure 2.16. It does not differ across different instances of the same object. For example, if an image contains two cars, semantic segmentation assigns the same label to all of the pixels in both cars. It provides dense pixel-by-pixel labeling of the image, resulting in scene comprehension. Semantic segmentation was once thought to be a difficult task. The development of accurate and efficient approaches was made possible with the help of fully convolutional neural networks (FCNs) [41]. The level of sophistication of semantic segmentation has recently increased rapidly, and so has the computational power of embedded systems, allowing for commercial deployment. Segmentation models are highly useful for autonomous cars as we need to provide

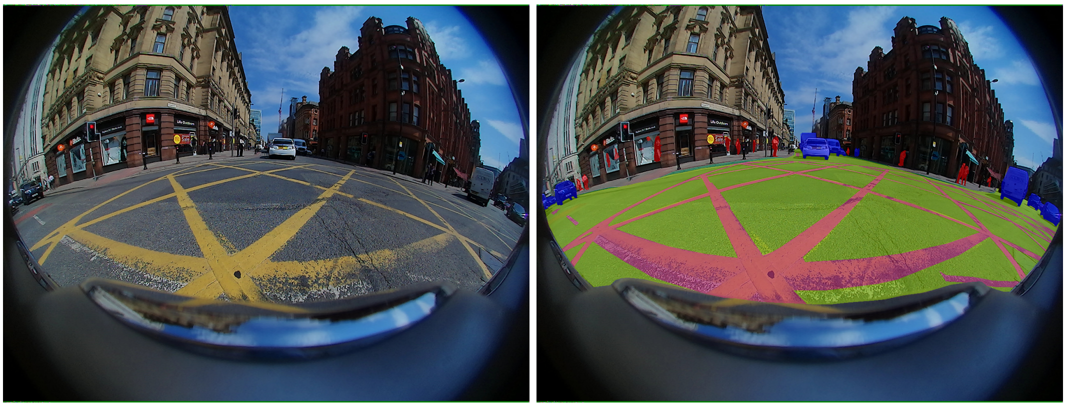


Figure 2.16 **A qualitative input sample (left) and semantic segmentation prediction (right) on WoodScape [12].**

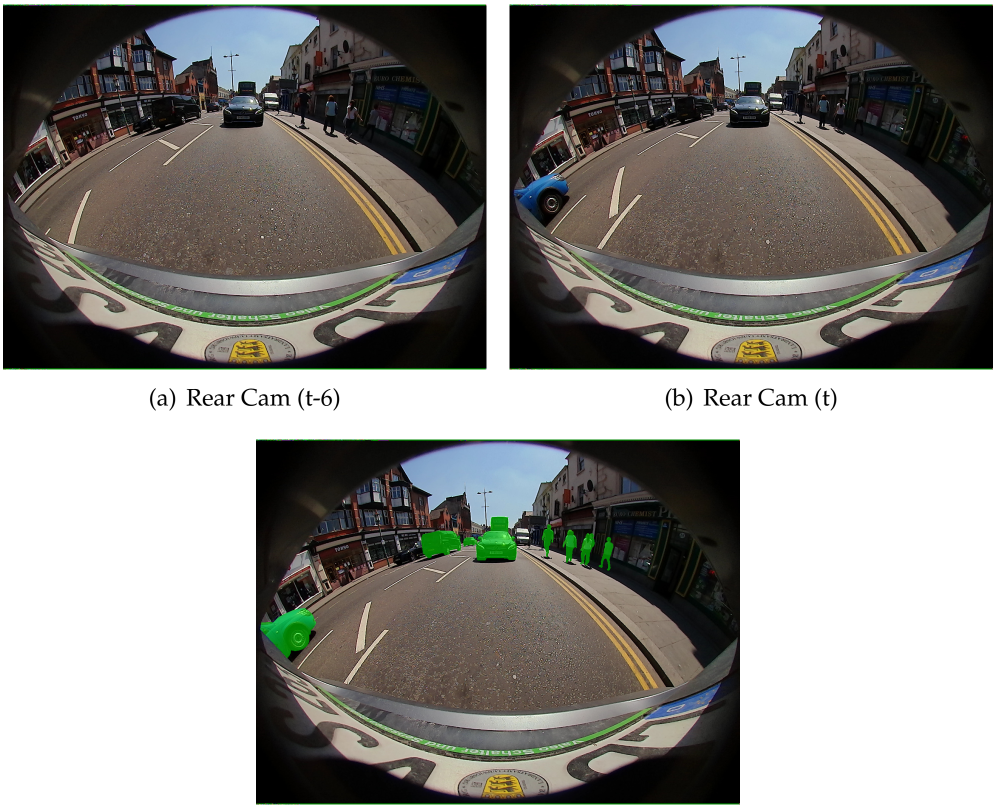
cars a detailed perception to enable them to understand their surroundings so that they can safely transition on our existing roads.

2.2.4 Motion Segmentation

Motion Segmentation is defined as the task of identifying the independently moving objects (pixels) such as vehicles and persons etc., in a pair of sequences and separating them from the static background as shown in Figure 2.17. It involves assigning a class label to each pixel in an image to segment static and dynamic objects. This task is treated as a binary segmentation problem. There are two types of motion in an autonomous driving scene. The first one is the motion of the surrounding obstacles and the second is the motion of the ego-vehicle. The ego-motion might cause difficulties to successfully detect the moving objects because even static objects will be perceived as moving. Motion segmentation implies two tasks that are performed jointly. The first one is object detection. We highlight the interesting objects only of specific classes, *i.e.*, pedestrians and vehicles, and discard any motion perceived from the background due to ego-motion. The second is motion classification, in which a binary classifier predicts whether the object is moving or static.

Because of the cameras' ego-motion on the moving vehicle, motion is a powerful cue in automotive driving, and detecting dynamic objects around the car is vital. Furthermore, it aids in detecting generic objects based on motion cues rather than appearance cues since there would still be unusual objects such as kangaroos or construction vehicles. The autonomous driving scenes are highly dynamic, where there are many moving objects interacting with each other forming a very complex environment to deal with. The detection and localization of moving obstacles are crucial for ADAS and autonomous vehicles. They are essential for emergency braking, support decision-making for its next step navigation, and for avoiding possible collisions [60]. An autonomous vehicle has to estimate collision risk with other interacting objects in the environment and calculate an optional trajectory. Collision risk is typically higher for moving objects than static ones due to the need to estimate the future states and poses of the objects for decision making. This is particularly important for near-range objects around the vehicle, which are typically detected by a fisheye surround-view system that captures a 360° view of the scene.

From a static observation point, the detection of moving obstacles is almost trivial as any non-zero optical flow will be due to motion in the scene or noise in the



(a) Rear Cam (t-6)

(b) Rear Cam (t)

(c) Motion Estimate

Figure 2.17 **A qualitative sample of motion segmentation on WoodScape. (t-6) and (t) frames are showcased to visually spot dynamic objects segmented in the motion estimate.**

image. For a moving observer, the problem is challenging as the entire scene relative to the camera moves. It is also complicated when considering fisheye cameras, which exhibit complex motion patterns due to the non-linear projection and strong lens distortion. Fewer classes are movable, and this can be leveraged to improve classification accuracy. For example, object classes such as buildings or poles are static and will not have dominant motion vectors after ego-motion compensation.

2.2.5 Soiling Segmentation

Level 3 autonomous driving [24] stands out as a challenging goal of a large part of the computer vision and machine learning community. Due to this problem's difficulty, a combination of various sensors is necessary to build a safe and robust system. However, apart from the geometric and semantic tasks, there are other less "popular" problems slowly getting into attention, which have to be solved for the ultimate goal of the full *Level* 5 autonomy. Cameras are an essential part of the sensor suite to achieve *Level* 3 autonomous driving. Surround-view cameras are, however, directly exposed to the external environment and are vulnerable to get soiled as a consequence of bad weather conditions such as rain, fog, or snow [43, 73]. Furthermore, dust and mud have a significant impact on vision tasks' performance. Cameras have a much higher degradation in performance due to soiling compared to other sensors. Thus it is critical to accurately detect soiling on the cameras, particularly for higher autonomous driving levels. As the visual perception modules for autonomous driving

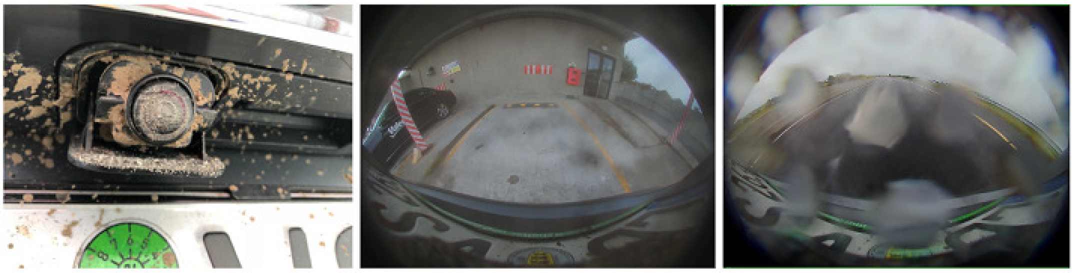


Figure 2.18 **Automotive surround-view cameras are exposed to harsh environmental setup.** Left: camera lens covered by mud. Middle: image produced by the soiled camera from the left picture. Right: camera lens soiled during heavy rain.

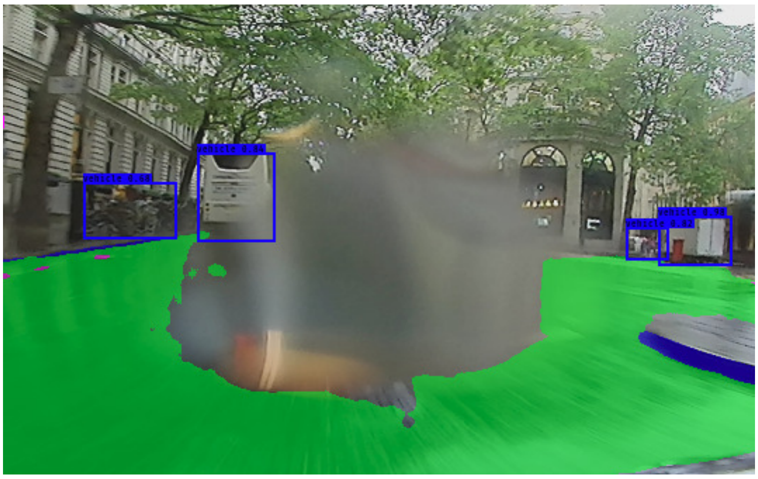


Figure 2.19 **Example of semi-transparent soiling in the form of a water drop on the camera lens.** The detection of the bus behind the water drop works still well, while the road segmentation (green) is highly degraded in the soiled region. In this scenario, a soiling detection algorithm is used to trigger a camera cleaning system that restores the lens hardware.

are becoming more mature, there is much recent effort to make them more robust to adverse weather conditions. It can be seen by the popular CVPR workshop "Vision for all seasons”[1], which focuses on the performance of computer vision algorithms during adverse weather conditions for autonomous driving.

One of these problems is the reliability of the sensory signal, which in the case of surround-view cameras means, *inter alia*, the ability to detect soiling on the camera lens. Failure to recognize severe weather conditions leading to a deterioration of the image quality to such a level that any further image processing is unreliable [13]. Figure 2.18 shows how the surround-view camera can get soiled and the corresponding image output, as well as an example of images taken during a heavy rain. It usually happens when the tire splashes mud or water from the road or wind, depositing dust on the lens. Figure 2.19 shows an example of the strong impact of a significant water drop on the camera lens for object detection and semantic segmentation tasks.

[1] https://vision4allseasons.net/

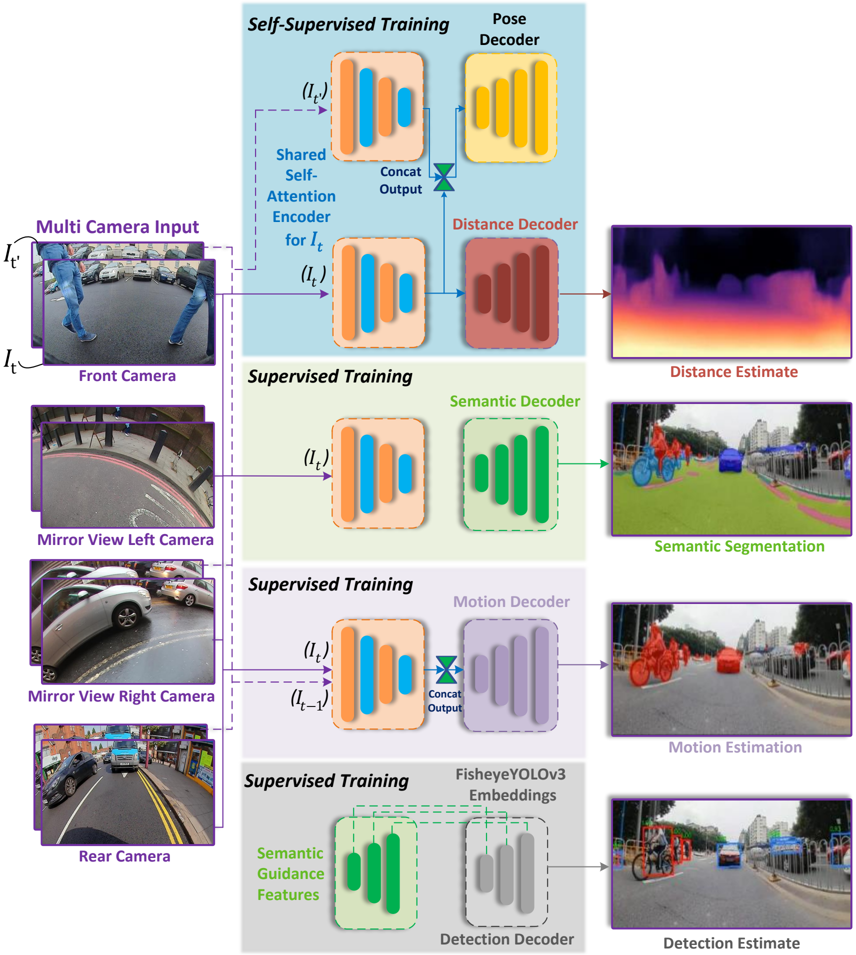


Figure 2.20 **Illustration of a multi-task architecture comprising of four tasks [1].**

2.3 Multi-Task Learning (MTL)

We usually care about optimizing a specific metric in deep learning, such as a score on a specific benchmark or a business key performance indicator. To accomplish this, we usually train a single model or a group of models to perform our desired task. The models are then fine-tuned and tweaked until their output no longer improves. While we can typically produce satisfactory results, we neglect details that might help us do even better on the metric we care about by being highly focused on a single task. We can derive knowledge explicitly from the training signals of related tasks. We may improve our model's generalization on our original task by sharing representations between similar tasks. This method is known as *Multi-Task Learning* (MTL) [129]. It is a sub-field of machine learning in which a shared model concurrently learns multiple tasks at once, as shown in Figure 2.20. The Figure 2.21 illustrates the output from OmniDet [1] MTL framework.

In general, when we find ourselves optimizing more than one loss function, we are engaging in MTL (in contrast to single-task learning). In such cases, it is beneficial to think about what we are attempting to do clearly in MTL and gain conclusions from it. Even if we are just optimizing one loss, as is usually the case, there is likely to be an auxiliary task that will assist us in improving our main task. MTL's target is

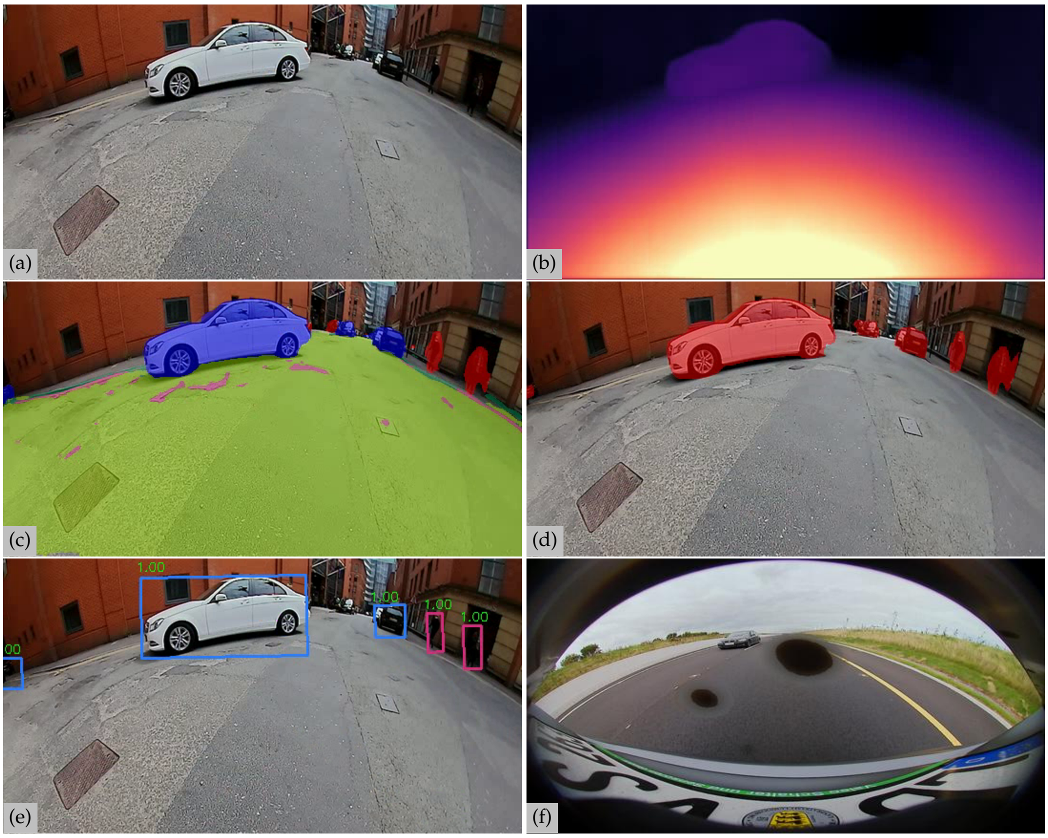


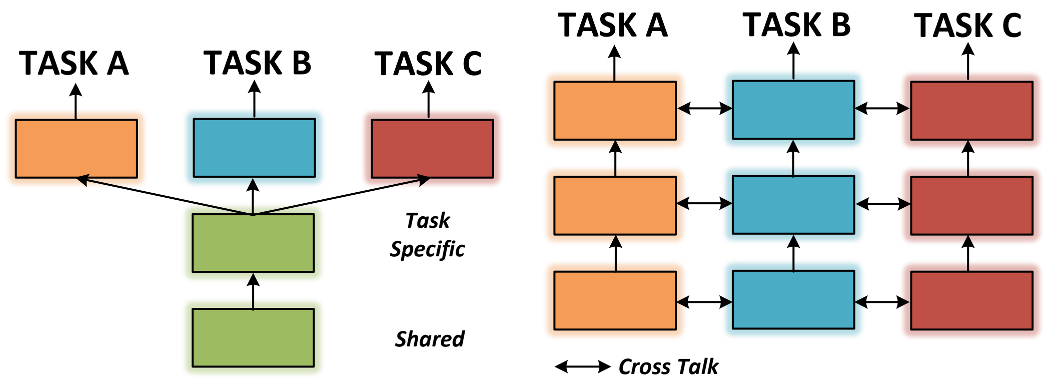
Figure 2.21 **Sample Multi Task Learning perception output on raw fisheye images.** (a) Rear-Camera Input Image, (b) Distance Estimate, (c) Semantic Segmentation, (d) Motion Estimation, (e) Standard Object Detection and (f) Soiling Segmentation.

succinctly stated by Rich Caruana [130]: "MTL enhances generalization by exploiting the domain-specific knowledge found in the training signals of similar tasks." MTL aims to solve computational bottlenecks in CNNs and increase computational performance by sharing the costly layers across all tasks.

2.3.1 Motivation

We may motivate MTL in various ways: It can be seen as biologically inspired by human learning. We often apply information gained from learning similar tasks while learning new tasks. An infant, for example, learns to recognize faces first and then applies this information to recognize other objects. From the standpoint of machine learning, we can inspire MTL by seeing it as a type of inductive transfer. Inductive transfer can help develop a model by adding an inductive bias that causes the model to favor some hypotheses over others. L_1 regularization, for example, is a common type of inductive bias that results in a preference for sparse solutions. In MTL, the auxiliary tasks provide the inductive bias, causing the model to favor hypotheses that describe more than one task [129].

From the automotive perspective in this thesis, our goal is to build a perception system that constitutes a **Level 3** autonomous stack. Deploying an efficient multi-task model has several advantages over all the earlier discussed single-task models. We could attain significant gain in embedded performance, certification, validation,



(a) Hard parameter sharing

(b) Soft parameter sharing

Figure 2.22 **Multi-task learning employing neural networks has been divided into soft and hard parameter sharing schemes.**

testing. Also, the deployment of a single MTL model is easier than several single-task learning (STL) models.

2.3.2 Two MTL approaches for Deep Learning

So far, we have concentrated on theoretical motives for MTL. To put MTL concepts into context, we consider the two most popular approaches for performing MTL in Deep Neural Networks (DNNs). Initially, MTL architectures were classified into *hard* or *soft* parameter sharing methods. We illustrate the hard parameter sharing in Figure 2.22 on the left, wherein the parameter set is divided into shared and task-specific parameters. The most popular approach to MTL in neural networks is hard parameter sharing, which goes back to [131]. MTL networks employing hard parameter sharing usually comprise a shared encoder that branches out into task-specific heads [132, 133, 134, 135, 136]. Overfitting is significantly reduced by hard parameter sharing. Baxter *et al*. [137] demonstrated that the risk of overfitting the shared parameters is an order N smaller than the risk of overfitting the task-specific parameters, *i.e.*, the output layers. This makes intuitive sense: the more tasks we learn simultaneously, the more work the model, has to put into finding a representation that captures all of the tasks, and the less risk we have of overfitting on our original task [129]. The first hard parameter sharing model was introduced by UberNet [138] wherein they tried to jointly tackle a large number of low, mid, and high-level perception tasks.

On the other hand, in soft parameter sharing, each task has its own model with its own set of parameters, and a feature sharing mechanism handles the cross-task talk (see Figure 2.22 (right)). The distance between the model's parameters is then regularized to allow the parameters to be similar. For example, Duong *et al*. [139] employs the L_2 norm for regularization, whereas Yang *et al*. [140] employs the trace norm. Cross-stitch networks [141] proposed soft-parameter sharing in MTL architectures [129].

2.4 Adversarial Attacks

Autonomous Vehicles are expected to significantly reduce accidents [142], where visual perception systems are in the heart of these vehicles. Despite the notable

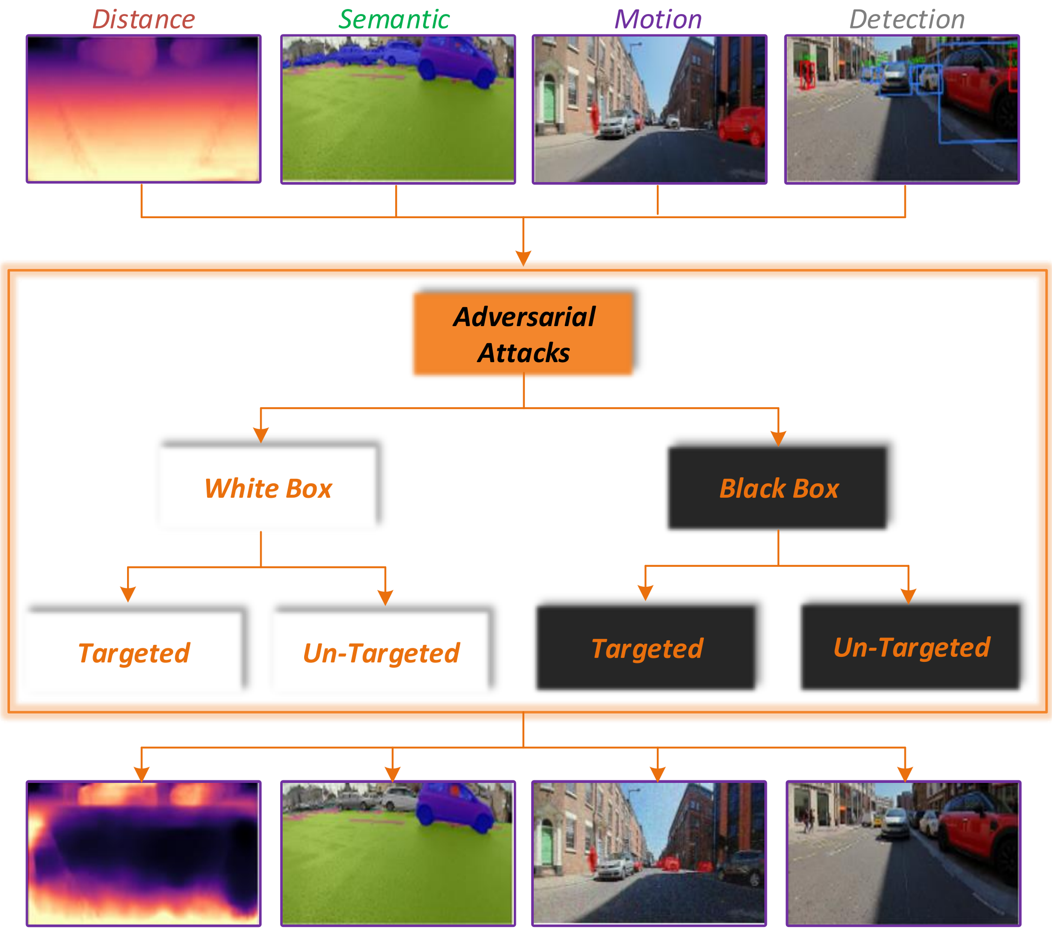


Figure 2.23 **Adversarial attacks on the OmniDet [1] MTL model.** Distance, segmentation, motion and detection perception tasks are attacked by white and black box methods with targeted and un-targeted objectives, resulting in incorrect model predictions.

achievements of DNNs in visual perception, we can easily fool the networks by adversarial examples that are imperceptible to the human eye but cause the network to fail. Hence, it is important to understand the robustness of MTL models over these attacks. Adversarial examples are usually created by deliberately employing imperceptibly small perturbations to the clean inputs, resulting in incorrect model outputs. This small perturbation is progressively amplified by a deep neural network (DNN) and usually yields wrong predictions. Generally speaking, attacks can be a white box or black box (see Figure 2.23) depending on the adversary's knowledge (the agent who creates an adversarial example). White box attacks presume full knowledge of the targeted model's design, parameters, and, in some cases, training data. Gradients can thus be calculated efficiently in white box attacks using the back-propagation algorithm. In contrast, in Black box attacks, the adversary is unaware of the model parameters and has no access to the gradients. Furthermore, attacks can be targeted or untargeted based on the intention of the adversary. Targeted attacks try to fool the model into a specific predicted output. In contrast, untargeted attacks consider the predicted output irrelevant, and the main goal is to fool the model into any incorrect output.

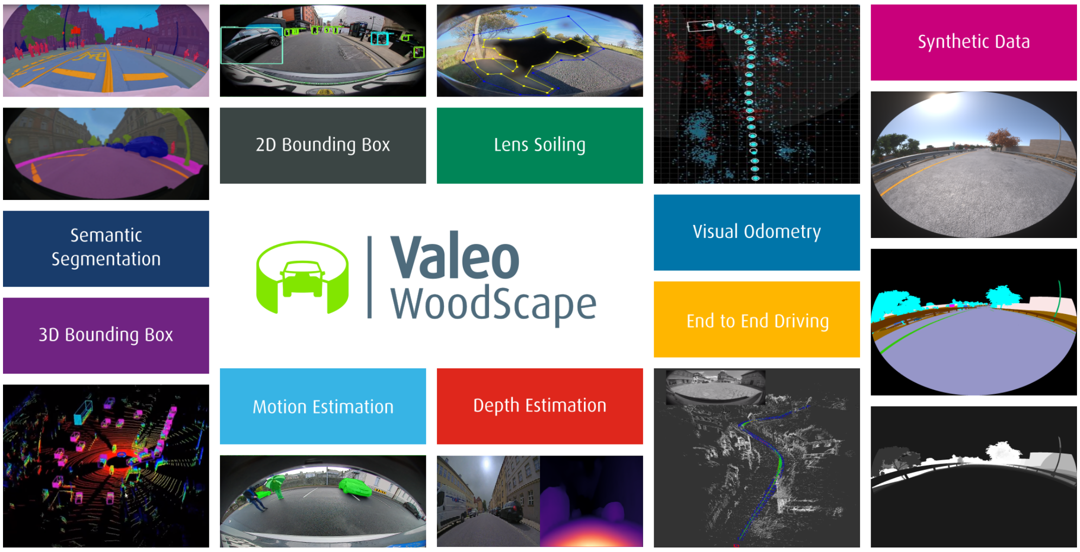


Figure 2.24 **Overview of the perception tasks in the WoodScape dataset.**

2.5 Datasets and Corresponding Benchmarks

In this section, we briefly discuss the datasets employed in this thesis for the experiments and the metrics used for the tasks.

2.5.1 WoodScape

The dataset consists of 46,000 images sampled roughly equally from the four views and split into training, validation, and test in a 6:1:3 ratio. This dataset is used for OmniDet [1] as explained in Chapter 6. A sub-set of 10,000 images from the dataset will be made public on **Github**[2]. A baseline code is released along with the dataset on **GitHub** to encourage further research to the community in developing unified perception models for autonomous driving. It contains several perception tasks listed in Figure 2.24. 2D box detection contains the five most essential categories of objects — *pedestrians, vehicles, riders, traffic signs, and traffic lights*. Vehicles further have subclasses, namely cars and large vehicles (trucks/buses). The polygon prediction task on raw fisheye is limited to only two classes — *pedestrians and vehicles*. Unlike traffic lights and traffic signs, these categories are non-rigid in nature and quite diverse in appearance, making them suitable for polygon regression. We sample 24 points with high curvature values from each object instance contour for the polygon regression task. Learning these points helps to regress better polygon shapes, as these points at high curvature define the shape of the object contours. Semantic segmentation comprises of 6 classes on *road, lanes, curbs, two-wheeled vehicles, vehicles, and persons*. The images are in RGB format with 1MPx resolution and 190° horizontal FoV. The dataset is captured in several European countries and the USA. For the experiments, we used only the vehicles' class. Further details about the dataset usage and demo code can be found on the WoodScape website https://woodscape.valeo.com.

[2] https://github.com/valeoai/WoodScape

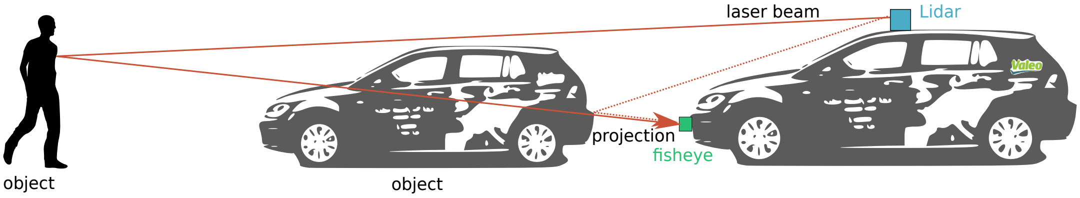


Figure 2.25 **The LiDAR's viewpoint lies higher than the fisheye camera's viewpoint.** 3D points from the object (person) will be mapped, even though – from the camera's point of view – the point is occluded.

WoodScape – Bamberg Dataset

The distance estimation dataset used in Chapter 3 and Chapter 4 for the work FisheyeDistanceNet [2], UnRectDepthNet [3], SynDistNet [4] and SVDistNet [5] contains roughly $40,000$ raw images obtained with multiple fisheye cameras constituting a surround-view system and point clouds from a sparse Velodyne HDL-64E rotating 3D laser scanner as ground truth for the test set. The training set contains $39,038$ images collected by driving around various parts of Bavaria, Germany. The validation and the test split contain $1,214$ and 697 images, respectively. The dataset distribution is similar to the KITTI Eigen split used in [53, 55] for the pinhole model and explained in detail in Section 2.5.2. The training set comprises three scene categories: *city*, *residential* and *sub-urban*. While training, these categories are randomly shuffled and fed to the network. We filter static scenes based on the vehicle's speed with a threshold of 2 km/h to remove image frames that only observe minimal camera ego-motion since distance cannot be learned under these circumstances. Comparable to previous experiments on pinhole SfM [53, 55], we set the length of the training sequence to 3.

Occlusion Correction

The data's sensor fusion will be correct if both camera and the Velodyne LiDAR scanner observe the world from the same viewpoint. However, in our vehicle, the fisheye cameras are in the front, and LiDAR is placed at the top, as seen in Figure 2.25. LiDAR perceives the environment behind objects that occlude the view of the camera. This problem of occlusion results in a wrong mapping of depth points that are not visible to the camera. It is harder to solve since occluded points are projected adjacently to disoccluded points.

To solve this problem, we adapted a distance-based segmentation technique from [10] with morphological filters as shown in the Figure 2.26. The depth image is denoted by d^{img}. Instead of directly mapping depth values on a single depth image d^{img}, we first split this image in I sub-windows $d_i^{\text{img}}, i = 1, \ldots, I$ of size 1280×800 $(W \times H)$ in the image plane, based on the lasers' distance projected from the LiDAR. Each d_i^{img} is mapped with depth-points in the form of intensity on the image plane in the increasing order of distance from the LiDAR's lasers. We apply a morphological filter that is dilation on each d_i^{img} to fill the sparse regions. The occluded points are removed with a thresholding technique *i.e.* $I^{\text{laser}} > I^{d_i^{\text{img}}}$ regarded as occlusion, where I^{laser} is the LiDAR's laser intensity based on distance and $I^{d_i^{\text{img}}}$ is intensity of dilated sub-windows d_i^{img} based on distance. The results are shown in Figure 2.28.

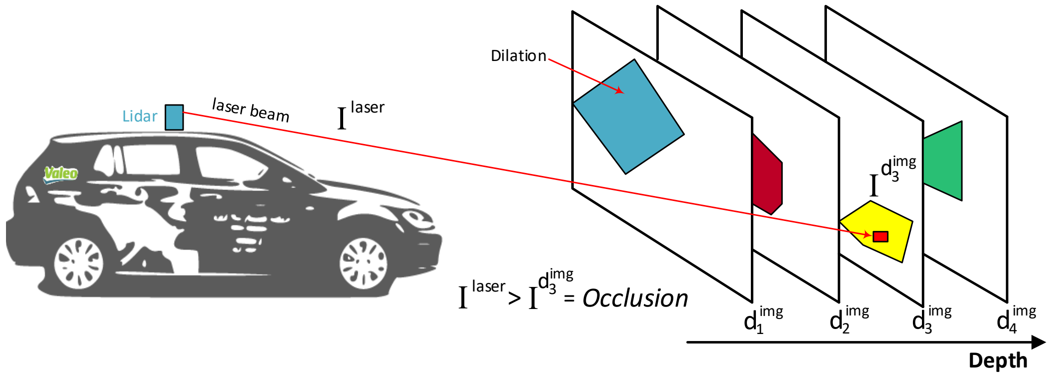


Figure 2.26 **Distance-based segmentation technique with morphological filters.** The sliced sub-windows of the single depth image d^{img}, contains intensity values in ascending order.

2.5.2 KITTI

The KITTI dataset consists of 42, 382 stereo sequences with corresponding raw LiDAR scans, 7, 481 images with bounding box annotations, and 200 training images with semantic annotations with a resolution of 1242 × 375. The unrectified dataset has a resolution of 1392 × 512. Geometric understanding tasks such as depth, flow, and pose estimation are widely benchmarked using KITTI. We use the data split according to Eigen *et al.* [51] for self-supervised depth estimation and filter the static frames as proposed by Zhou *et al.* [53]. The resulting training and validation set contains 39, 810 and 4, 424 monocular triplets. We use the standard test set of 697 images, which covers a total of 29 scenes. We use a single camera intrinsic matrix for the entire training and validation set. The camera's principal point is centered, and the focal length is set as the average of all the focal lengths in KITTI. The length of the training sequence is set to 3. We also use the 652 test frames from the Eigen split with improved ground truth provided by [143]. We utilize the KITTI split [54], whose test set includes the 200 training images from the KITTI 2015 Stereo dataset [144]. This test set's advantage is that it contains labels for depth and semantic segmentation, suitable to ablate the benefits of MTL.

2.5.3 Cityscapes

Cityscapes dataset has 2, 975 training and 500 validation images, with a resolution of 2048 × 1024 captured in 50 different cities. It features higher resolution street images of higher quality compared to KITTI. It has a similar setting compared to KITTI but contains more dynamics scenes. The validation set in principle provides ground-truth labels for depth estimation and semantic segmentation tasks; the depth labels are obtained by a classical Semi-Global Matching [145] algorithm. The KITTI dataset's depth labels are physical measurements from a LiDAR sensor and thereby better suited for evaluating a depth estimation model. For pixel-wise semantic segmentation, the dataset contains 19 classes. We extracted the 2D boxes from the instance polygons for the entire training and validation split.

2.6 Evaluation

This section explains the metrics used for all the visual perception tasks in this thesis.

2.6.1 Metrics for Depth Estimation

Abs Rel : $\frac{1}{\lvert N \rvert} \sum_{i \in N} \frac{\lvert d_i - d_i^* \rvert}{d_i^*}$	**RMSE** : $\sqrt{\frac{1}{\lvert N \rvert} \sum_{i \in N} \parallel d_i - d_i^* \parallel^2}$
Sq Rel : $\frac{1}{\lvert N \rvert} \sum_{i \in N} \frac{\lVert d_i - d_i^* \rVert^2}{d_i^*}$	**RMSE log** : $\sqrt{\frac{1}{\lvert N \rvert} \sum_{i \in N} \parallel \log(d_i) - \log(d_i^*) \parallel^2}$
Accuracies δ_t : $\frac{1}{\lvert N \rvert} \lvert \{ d \in N \rvert \max(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}) < 1.25^t \} \rvert \times 100\%$	

Table 2.2 **P**erformance indicators for depth evaluation. where d_i and d_i^* denotes the ground truth and predicted depth value of pixel i, respectively. N denotes the set of pixels with real-depth/distance values in an image, $|.|$ returns the number of the input set elements.

Table 2.2 indicates the performance indicators for depth evaluation. Saxena *et al.* [146, 147] proposed the first set of criteria defined to assess the quality of an estimated depth map. It consists of computing the percentage of pixels having a relative error δ lower than a certain threshold. A larger set of metrics were proposed by Eigen *et al.* [50] for depth prediction given by:

- **Absolute Relative error (Abs Rel)**: Normalizes per-pixel errors according to real depth, reducing the effect of large errors with the distance. It ignores the difference's sign, preventing positive and negative errors from canceling each other out. The error averaged over the entire test set is the score given in the literature. The metric is not specified for null measurements, which is irrelevant since depth is never zero.

- **Squared Relative error (Sq Rel)**: The squared term penalizes larger depth errors (*e.g.*, near discontinuities). It is expressed likewise to the absolute relative error, except it has the effect of penalizing particularly bad predictions. In the case of Abs Rel, the symbol of the difference is ignored. Differences larger than one have a greater impact on Sq Rel than on Abs Rel, while differences smaller than one have a greater impact on Abs Rel than on Sq Rel. As a result, comparing the two tests should aid in determining whether the errors are caused by outliers or by several small offsets in the forecast. Sq Rel should not be used in isolation because the existence of outliers can lead to incorrect conclusions about a model's results. It is measured in the same units as the data.

- **Root Mean Squared Error (RMSE)**: A traditional metric for measuring regression errors. It measures the standard deviation of the error; a higher value implies widely varying prediction accuracy. The RMSE is defined as the squared root of the mean quadratic difference between the prediction and the ground truth. The RMSE, like Sq Rel, is susceptible to outliers. The squared root is used to ensure that the RMSE is expressed in the same unit as the data.

- **Root Mean Squared logarithmic error (RMSE log)**: The logarithm makes this error relative, reducing the effect of large errors with the distance. It is more invariant to the scale of the error and is meriting to note that it penalizes underestimates more than overestimates, which makes it less valuable of a metric for some applications of depth estimation, like autonomous navigation.

- **Three inlier thresholds** are used to determine the quality of the depth predictions. These are the percentage of predictions within some factor δ of the ground truth. The standard measures, used in these experiments, are $\delta < 1.25^t, t = \{1, 2, 3\}$. In other words, a value of 0.9 for δ_1 means that 90% of the pixels are such that the difference between the ground truth d_i and the prediction d_i^* is less than 25% of the smallest of the two (*i.e.*, 25% of the prediction if $d_i < d_i^*$ and conversely). Contrary to previous metrics, the model performs better when the accuracy is higher.

2.6.2 Metrics for Segmentation-Based Tasks

Some basic concepts used by the metrics:

- True Positives (TP): A true positive is an outcome where the model correctly predicts the *positive class i.e.*, a correct Segmentation. Segmentation with IOU $\geq$ threshold.
- False Positives (FP): A false positive is an outcome where the model incorrectly predicts the *positive class i.e.*, a wrong Segmentation. Segmentation with IOU $<$ threshold.
- False Negatives (FN): A false negative is an outcome where the model incorrectly predicts the *negative class i.e.*, a ground truth not segmented.
- True Negative (TN): A true negative is an outcome where the model correctly predicts the *negative class*.

Pixel Accuracy (PA): Also known as global accuracy [42] is a straightforward metric that measures the ratio between the number of correctly classified pixels and the total number of pixels. The mean pixel accuracy (mPA) metric computes the proportion of right pixels on a per-class basis. mPA is also known as *class average accuracy* [42]. Accuracy is obtained by taking the ratio of correctly classified pixels with respect to the total pixels. The key drawback of employing this metric is that the outcome may appear favorable if one class outnumbers the other. If, for example, the background class covers 90% of the input image, we can achieve 90% accuracy by simply classifying every pixel as background.

$$PA = \frac{\sum_{j=1}^{k} n_{jj}}{\sum_{j=1}^{k} t_j}, \qquad mPA = \frac{1}{k}\sum_{j=1}^{k}\frac{n_{jj}}{t_j} \tag{2.27}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.28}$$

where n_{jj} represents the total number of pixels classified and labeled as class j. In other terms, n_{jj} is the total number of *True Positives* for class j. The total number of pixels labeled as class j is t_j [148].

Intersection over Union (IoU): Also known as the Jaccard Index, the IoU statistic compares the similarity and diversity of sample sets. It is the ratio of the intersection of the pixel-wise classification results with the ground truth to their union as shown in Figure 2.27.

$$IoU = \frac{\sum_{j=1}^{k} n_{jj}}{\sum_{j=1}^{k}(n_{ij} + n_{ji} + n_{jj})}, \qquad i \neq j \tag{2.29}$$

where n_{ij} denotes the number of pixels labeled as class i but listed as class j. They are, in other terms, *False Positives* (false alarms) for class j. Similarly, n_{ji}, the total number of pixels labeled as class j but listed as class i, represents the *False Negatives* (misses) for class j [148]. An extended version of IoU is widely in use. It is given by:

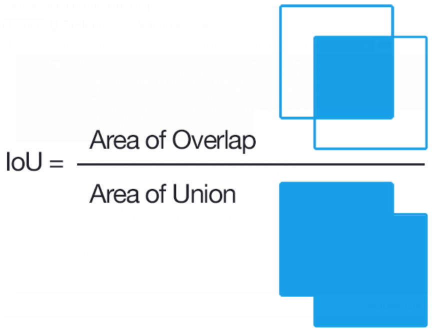


Figure 2.27 **Depiction of Intersection over Union (IoU) metric.** Figure reproduced from [149].

Mean Intersection over Union (mIoU): mIoU is the class-averaged IoU, *i.e.*

$$mIoU = \frac{1}{k}\sum_{j=1}^{k}\frac{n_{jj}}{n_{ij}+n_{ji}+n_{jj}}, \qquad i \neq j \tag{2.30}$$

2.6.3 Metrics for Object Detection

Evaluation in object detection is difficult since there are two distinct tasks to measure:

- Determining the presence or absence of an object in the scene (classification).
- Determining the object's position (localization, a regression task).

Furthermore, there will be several classes in a standard data set, and their distribution will be non-uniform (*e.g.*, there might be many more vehicles than traffic signals). As a result, biases would be introduced by a simplistic accuracy-based metric. It is also essential to determine the possibility of misclassifications. As a result, a "confidence score" or model score must be assigned to each bounding box observed, and the model must be evaluated at different levels of confidence. The Average Precision (AP) was created to meet these requirements. To comprehend the AP, one must first understand the Precision and Recall of a classifier [150].

Precision is defined as the "False Positive Rate," or the ratio of true object detections to the total number of objects predicted by the classifier. If the precision score is close to 1, there is a good chance that whatever the classifier predicts as a positive detection is accurate. In other terms, Precision is a model's ability to identify only relevant objects. It is calculated as a percentage of correct positive predictions *i.e.*,

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} = \frac{\text{TP}}{\text{all detections}} \tag{2.31}$$

The "False Negative Rate," or the ratio of true object detections to the total number of items in the data collection, is measured by *Recall*. If the recall score is close to 1, the model can correctly detect almost all the objects in the dataset. In other terms, the ability of a model to identify all relevant cases is referred to as Recall (all ground truth bounding boxes). It is given as the percentage of true positives detected among all related ground truths and is calculated as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{\text{TP}}{\text{all ground truths}} \tag{2.32}$$

Finally, it is essential to remember that Precision and Recall have an inverse relationship. These metrics are affected by the model score threshold being set and the quality of the model. To compute the Average Precision (AP), the precision-recall curve for a specific class (*e.g.*, car) is calculated from the model's detection output by varying the model score threshold that defines what is counted as a model-predicted positive detection of the class [150].

Average Precision (AP): The method of computing AP by the PASCAL VOC challenge has improved since 2010. Currently, the PASCAL VOC challenge interpolation uses all data points, rather than interpolating just 11 equally spaced points (11-point interpolation method) as described in [151], and they propose to *interpolate all data points*.

11-point interpolation: The 11-point interpolation tries to summarize the shape of the Precision × Recall curve by averaging the precision at a set of eleven equally spaced recall levels $[0, 0.1, 0.2, \ldots, 1]$:

$$\mathrm{AP} = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \rho_{\text{interp}(r)} \tag{2.33}$$

with

$$\rho_{\text{interp}} = \max_{\tilde{r}: \tilde{r} \geq r} \rho(\tilde{r}) \tag{2.34}$$

where $\rho(\tilde{r})$ is the measured precision at recall $\tilde{r}$. Instead of using the precision observed at each point, the AP is calculated by interpolating the precision only at the 11 levels r taking the **maximum precision whose recall value is greater than** r [152].

Interpolating all points: Instead of just interpolating between the 11 equally spaced points, we might interpolate between all n points in such a way that:

$$\sum_{n=0} \left(r_{n+1} - r_n\right) \rho_{\text{interp}} \left(r_{n+1}\right) \tag{2.35}$$

with

$$\rho_{\text{interp}} \left(r_{n+1}\right) = \max_{\tilde{r}: \tilde{r} \geq r_{n+1}} \rho(\tilde{r}) \tag{2.36}$$

where $\rho(\tilde{r})$ represents the calculated precision at recall $\tilde{r}$. Instead of using the precision observed at just a few points, the AP is now calculated by interpolating the precision at each step, with r taking the maximum precision whose recall value is greater or equal to $r + 1$. We measure the approximate region under the curve in this manner [152].

Localization and Intersection over Union: To assess the model's performance on the task of object localization, we must first determine how well the model predicted the object's position. Typically, this is accomplished by drawing a bounding box around the object of interest, but in some situations, an N-sided polygon or even pixel by pixel segmentation is used. The localization task is usually evaluated on the Intersection over Union threshold in both of these situations (IoU). In this thesis, for fisheye images, the object representation is mainly carried out using N-sided polygons. Therefore, we widely use (mIoU) as shown in Eq. 2.30.

Finally, now that we have defined AP and IoU thresholds, the mean Average Precision (mAP) score is computed by averaging the AP across all classes and/or IoU thresholds.

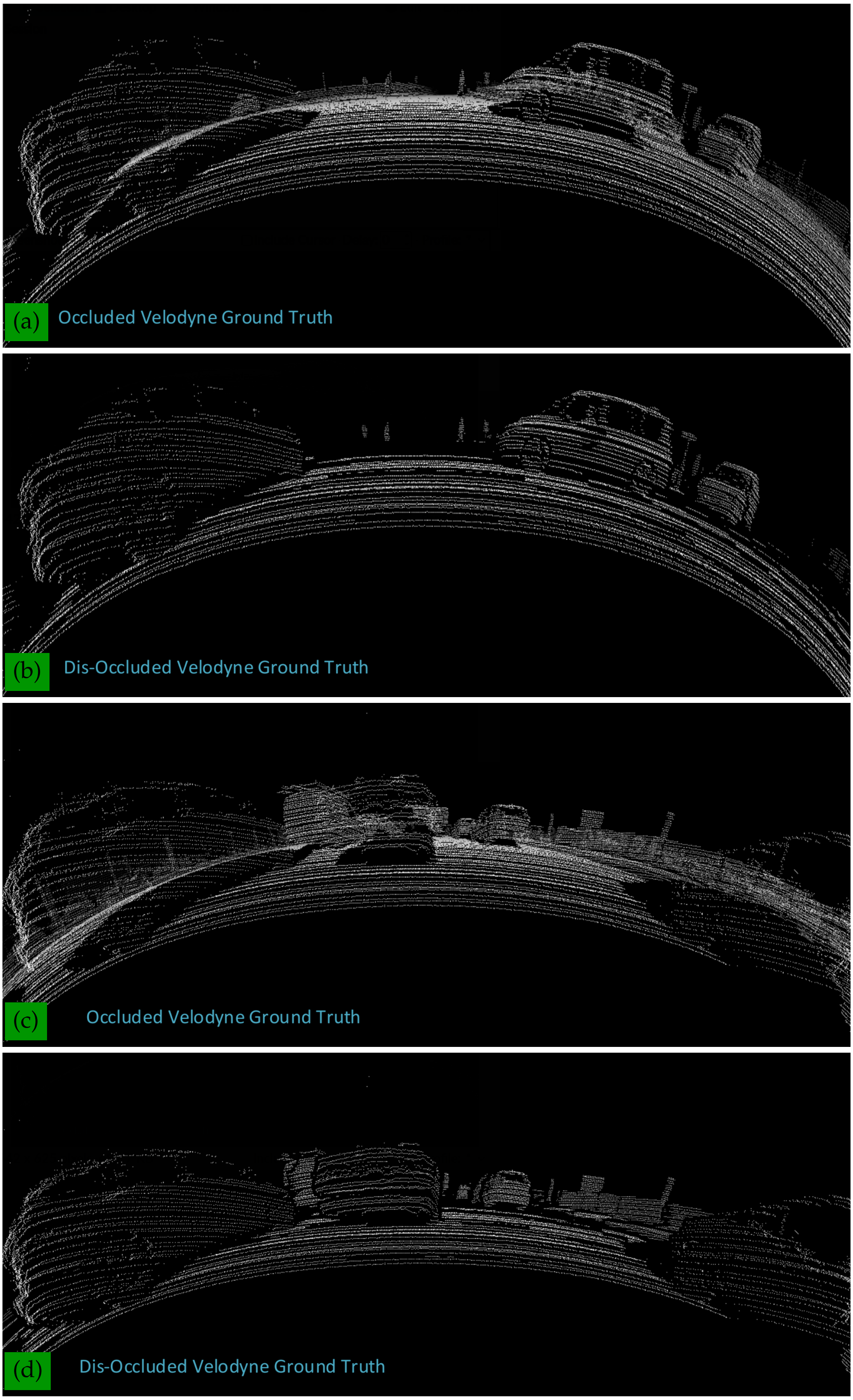

(a)
Occluded Velodyne Ground Truth
(b)
Dis-Occluded Velodyne Ground Truth
(c)
Occluded Velodyne Ground Truth
(d)
Dis-Occluded Velodyne Ground Truth

Figure 2.28 **Ground truth LiDAR values projected onto an image to obtain distance maps.** (a) and (c) Occluded LiDAR ground-truth maps, (b) and (d) Dis-Occluded LiDAR ground-truth maps

Chapter 3

Related Work

Contents

3.1 Depth Estimation

Numerous deep learning models to infer scene depth from images have been proposed in recent years. They differ in many ways, including architecture and layer design, the required number of input images, training strategy, and datasets on which they are trained and tested. Among the various points used to categorize models, the number of images and the chosen training strategy have a strong influence on the quality of estimated depth maps, while the architecture design may affect the model's ability to generalize on unseen data [153].

When the training technique is used as a criterion for classification, three distinct classes emerge. To learn the task of depth prediction, supervised methods include ground truth depth maps associated with RGB inputs. Then there are unsupervised methods, which require only RGB images during training, and semi-supervised methods, which combine the two approaches and depend on ground truth depth maps when they are available in the training set. Models that are supervised are known to produce better results, but they come at a cost. As discussed in [154], collecting datasets containing both RGB images and depth maps associated with them is difficult and error-prone. As a result, such models are often trained on a small number of datasets, restricting both research and implementation possibilities [153].

Methods are classified into the same two groups as non-learning approaches for the amount of RGB images retained, namely monocular and multi-view algorithms. The former can only infer depth from a single image, whereas the latter requires at least two. Monocular methods are less accurate. While some agree that the difference between the two types of algorithms can be bridged, others claim that monocular approaches can never catch up because there is no way to compensate for the lack of detail when only one image is used. Furthermore, monocular depth estimation is an ill-posed problem, meaning that any number of 3D structures will result in the same 2D observation. Nonetheless, applying smoothness constraints always yields acceptable results, often at the expense of fine details. Monocular algorithms produce up-to-scale depth maps, while multi-view methods can produce metric depth by depending on known real-world measurements such as the distance between two cameras in the recording setup [153].

These two characteristics are used in the following subsections to distinguish and present a broad range of state-of-the-art models.

3.1.1 Supervised Monocular Methods

Depth estimation is a crucial task for automated driving, and multi-view geometric approaches were traditionally used for computing depth. Some of the initial prototypes of automated driving relied primarily on depth estimation [155], and to enable accurate depth estimation, stereo cameras were used. As LiDAR-based depth perception is sparse and costly, image-based methods are of significant interest in perception systems regarding coverage density and redundancy. Eigen *et al*. [50] was one of the first to successfully prove that CNN's are capable of predicting depth from single images. Following its success, Eigen *et al*. [51] proposed a general multi-scale system that can handle tasks like depth map estimation, surface normal estimation, and semantic label prediction from a single image. Demon [156] is a network developed by Ummenhofer *et al*. that can predict depth and egomotion, *i.e.*, camera motion, from a pair of images taken by a single camera, thus solving the two-view

SfM problem. Vijayanarasimhan *et al*. [157] proposed a different architecture called SfM-Net to solve the same *SfM* problem. Their model can compute depth, egomotion, 3D rotations, and translations for the dynamic objects in the scene and motion masks for these dynamic objects, given a pair of images and camera intrinsics. SfM-Net is unique in that it can be used for a variety of training scenarios. It can be trained supervised, using egomotion or depth, or self-supervised, by learning its tasks without having access to ground truth data for the quantities it predicts. To infer depth from a single image, Liu *et al*. [158, 159] combined a convolutional network and a continuous conditional random field (CRF) [160]. Meanwhile, they suggested a super-pixel pooling method to speed up the CNN, and it aids in the design of the deeper network to increase depth estimation accuracy. Shelhamer *et al*. [161] proposes a fully connected network (FCN) framework for monocular depth estimation, after which the proposed framework jointly optimizes the intrinsic factorization to recover the input image. Laina *et al*. [162], inspired by ResNet's [163] outstanding performance, used residual learning to learn the mapping relation between depth maps and single images, resulting in a network that is deeper and more accurate than previous works in depth estimation. Furthermore, up-sampling blocks replace fully connected layers in ResNet to increase the resolution of the projected depth map. Chen *et al*. [164] investigated a novel algorithm to tackle the problem of perceiving single-image depth estimation in the wild and also learn the camera intrinsics. Rather than using the supervised signal of depth as the ground truth, their networks are trained using relative depth annotations.

Methods based on adversarial learning: In recent years, the adversarial learning proposed in [165] has become a highly investigated research path due to its outstanding output on data generation [166]. A broad range of algorithms, theories, and applications have been developed, as summarized in [167]. Stack GAN [168], Conditional GAN [169], and Cycle GAN [170] all based on [165], are incorporated into depth estimation tasks and have a positive effect on the depth estimation [171, 172, 173]. Jung *et al*.; incorporate adversarial learning into monocular depth estimation tasks in [172]. Here, a Global Net and a Refinement Net make up the generator, and these networks are designed to estimate global and local 3D structures from a single image. The predicted depth maps are then distinguished from the real ones using a discriminator, which is a typical type in supervised methods. For monocular depth estimation, conditional GAN is also used in [173]. The difference from [172] is that a secondary GAN is used to produce a more refined depth map based on the image and a coarse approximate depth map.

The supervised methods can effectively learn a function to map 3D structures, and their scale details since the ground truth supervises them. However, the labeled training sets, which are difficult and costly to obtain, restrict the application of these supervised methods [174].

3.1.2 Self-Supervised Monocular Methods

Although supervised methods learn their tasks efficiently using simple loss functions and typically produce good results, they also place strict limits on the datasets that can be used to train them. After all, they rely on ground truth depth maps that are compatible with the corresponding RGB images, which is difficult to achieve without errors. Furthermore, depending on the data acquisition system, the depth maps can be extremely sparse and contain numerous gaps, making learning challenging, if not impossible, in these areas. Self-Supervised approaches, on the other hand, are trained solely on RGB images and do not require any depth data, enabling training on a more significant number of datasets or making the recording of new ones easier [153]. Here, current state-of-the-art approaches rely on neural networks [175, 176], which can even be trained in an entirely self-supervised fashion from sequential images [53], giving a clear advantage over supervised approaches in terms of applicability to arbitrary data domains.

The approaches of Garg *et al.* [52], and Zhou *et al.* [53] showed that it is possible to train networks in a self-supervised fashion by modeling depth as part of a geometric projection between stereo images and sequential images, respectively. A model based on view synthesis from a pair of rectified stereo images has also been proposed by Godard [54]. In two main ways, their solution, Monodepth, varies from that of Garg [52]. To avoid the Taylor approximation, which makes optimization difficult, they produce images using bilinear sampling, resulting in a fully sub-differentiable training loss. Second, they enforce a left-right consistency check in the form of an additional term in the training loss to resolve the ill-posed nature of monocular depth estimation. Many stereo algorithms use this form of consistency check as a post-processing stage. However, they were able to integrate it directly into the network, allowing for end-to-end learning. Zhou *et al.* [53] suggested a paradigm that overcomes the constraint on potentially accessible datasets. Instead of stereo images for training, it employs consecutive frames captured by a moving camera. Aside from disparity, the model must also estimate camera motion between frames. Camera motion is required by the models of Garg, as well as Monodepth [54]. However, it is constant and known ahead of time because the stereo cameras are calibrated before recording the dataset. The only details needed are the camera's intrinsic parameters. The core concept remains that of view synthesis, as well as the same bilinear sampling process used in Monodepth [54].

The model of Mahjourian *et al.* [154] is designed to be trained on a broad range of datasets, similar to that of Zhou *et al.* [53]. Mahjourian *et al.* [154] predicts depth and egomotion, but it also considers explicitly the assumed 3D structure of the entire scene, rather than depending solely on a local photometric loss. Unlike the masks predicted by a dedicated network by Zhou [53], those of Mahjourian [154] are computed analytically, making the overall learning problem more straightforward. Wang *et al.* [177] discovered that unsupervised monocular models trained on monocular sequences, and constrained by the nature of the problem such as those used by Zhou *et al.* [53], do not perform as well as unsupervised monocular models trained on rectified stereo datasets, such as those used by Godard *et al.* [54]. To clarify the performance gap, they established two significant differences between stereo and monocular strategies: (i) unknown camera pose between frames and (ii) the uncertainty in scale inherent in all monocular models trained on monocular sequences. Both are only partly resolved by the pose network, and Wang *et al.* [177] demonstrated that scale uncertainty induces divergence during training. This is due to the scale sensitivity of the depth regularization terms (*i.e.*, the smoothing terms) used in the loss function, as shown empirically in their work. Furthermore, they argue that the pose network is superfluous and that it can be replaced by a differentiable and deterministic goal for pose prediction, as well as a simple normalization technique, both of which are widely used in Direct Visual Odometry (DVO) [153].

Indeed, pose estimation from depth is a well-studied problem with geometric properties and efficient algorithms. There are three benefits to replacing the pose network with one of them. It does not require any learning parameters, for starters, making the model simpler than those that rely on a pose network. Second, it establishes a direct relationship between the input dense depth map and the output pose prediction. In contrast, pose network models, except for Mahjourian *et al*. [154], ignore scene geometry and produce camera pose estimates based solely on photometric appearance. Third, given a depth estimate, DVO solves for camera pose by minimizing the same view synthesis loss used to train the network, preventing an increase in computations. Wang *et al*. [177] used a differentiable DVO (DDVO) algorithm similar to the inverse compositional spatial transformer network to enable backpropagation during training. Since it is a second-order gradient descent process, a good initialization point will lead to a better solution than one chosen at random. As a result, rather than starting from the identity pose, relying on a first estimate provided by a pre-trained pose network is likely to produce better results [153].

The initial concept has been extended by considering improved loss functions [54, 55, 178, 179, 180, 181], the application of generative adversarial networks (GANs) [178, 182, 183], generated proxy labels from traditional stereo algorithms [184], or synthetic data [185]. Other approaches proposed to use specialized architectures for self-supervised depth estimation [7, 177, 186], they apply teacher-student learning [187] to use test-time refinement strategies [188, 189], to employ recurrent neural networks [190, 191], or to predict the camera parameters [192] to enable training across images from different cameras.

3.1.3 Depth Estimation on Fisheye Cameras

Recent approaches also investigated the application of self-supervised depth estimation to 360° images [193, 194]. Most of the works have solely focused on traditional 2D content captured with cameras following a typical pinhole projection model based on rectified image sequences. Omnidirectional (360°) content is now more easily and consistently produced thanks to the development of efficient spherical cameras and rigs and is seeing increased adoption in robotics and autonomous vehicles. Many real-world applications rely on more advanced camera geometries *e.g.*, fisheye camera images.

With the surge of efficient and cheap wide-angle fisheye cameras and their larger FoV in contrast to pinhole cameras, there has been significant interest in the computer vision community to perform depth estimation from omnidirectional content similar to traditional 2D content via omnidirectional stereo [195, 196, 197]. There is also a trend of integrating depth estimation tasks into multi-task models [198, 199]. Most of the depth estimation methods were demonstrated in automated driving on rectified KITTI video sequences where barrel distortion was removed. The same multi-view geometry [200] principles apply to 360° images equivalently as they apply to pinhole camera images. The underlying geometrical structure can be estimated by observing the scene from multiple perspectives and establishing correspondences between them. By taking into account the different projection models and defining the disparity as angular displacements, the traditional binocular or multi-view stereo [201] problem is reformulated to binocular or multi-view spherical stereo [202] for 360° cameras. It was recently [203] demonstrated that using *SfM*, 360° videos captured with a moving camera can be used to reconstruct a scene's geometry.

Current CNN processing pipelines can be applied to spherical input in two simple ways. Either directly on a projected (usually equirectangular) image or by projecting spherical content to the faces of a cube (cube map) and running CNN predictions on them, which are then merged by back-projecting them to the spherical domain. New techniques for applying CNNs to omnidirectional input have recently been presented. Given the difficulty of directly modeling the projection's distortion in typical CNNs while also achieving invariance to the viewpoint's rotation, [204] proposes a graph-based deep learning approach. Su *et al*. [205] used a planar CNN to process 360° images in the equirectangular projection. They designed a novel approach by transferring appropriate convolution weights from an existing network trained on traditional 2D images to learn appropriate convolution weights for equirectangular projected spherical images. This conversion from the 2D to the 360° domain is achieved by enforcing consistency between the predictions of the 2D projected views and those in the 360° image. Recent work on convolutions [206, 207] that learn their shape, as well as their weights, has been applied to fisheye lenses [208]. However, apart from these works, applying self-supervised depth estimation to more advanced geometries, such as fisheye camera images, has not been investigated extensively yet.

Compared to the state-of-the-art approaches, in this thesis, we explore the *SfM* approach to developing a self-supervised training strategy that aims to infer a distance map from a sequence of distorted and unrectified raw fisheye and pinhole images. We aim to develop a generic end-to-end self-supervised training pipeline to estimate monocular depth maps on raw distorted images for various camera models. We create a training framework for self-supervised distance estimation, which jointly trains and infers images from multiple fisheye cameras and viewpoints. Also, improve the photometric loss by a general and robust loss function. Further, we explore the avenues in introducing a novel architecture for the learning of self-supervised distance estimation synergized with semantic segmentation. We look upon the issues of the dynamic object impact on self-supervised distance estimation by using semantic guidance.

3.2 Object Detection

We can broadly classify the state-of-the-art object detection methods based on deep learning into two types: two-stage detectors and single-stage detectors. We provide an overview of the categorization of CNN-based object detection methods for object detection on pinhole camera images in Table 3.1. We categorize based on the number of stages involved in the framework, deformable part-based detection methods, and keypoint-based detectors.

3.2.1 Object Detection on Pinhole Cameras

Two-Stage Detectors

The object detection task is divided into two stages in this approach: (i) extraction of Regions of Interest (ROIs) and (ii) classification and regression of the ROIs. Regions with CNN features (R-CNN) by Girshick *et al*. [209] was the first to use a two-stage approach. It generates ROIs through selective search and classifies ROIs using a DCN-based classifier. It requires complex computations, which causes it to be slow and far from real-time capable. By extracting ROIs from feature maps, Fast R-CNN [37] and SPP-Net [223] improved R-CNN [209]. SPP-Net used a Spatial Pyramid Pooling (SPP) layer to manage images of arbitrary sizes and aspect ratios. It applies an SPP layer over the feature maps produced by convolution layers and produces the fixed-length vectors required by fully connected layers. It removes the need for fixed-size inputs

CNN based Object Detection methods				
Single stage detectors	*Two stage detectors*	*Multi-stage detectors*	*Deformable part based detectors*	*Keypoint based detectors*
SSD [125]	R-CNN [209]	Cascade R-CNN [210]	DPM-CNN [211]	CornerNet [212]
Yolo9000 [38]	Fast R-CNN [37]	CRAFT [213]	DeepIDNet [214]	ExtremeNet [215]
Retinanet [216]	Faster R-CNN [126]	CC-Net [217]	DP-FCN [218]	CenterNet [219]
Squeezedet [220]	R-FCN [221]	Multipath Net [222]	Deformable ConvNets [207]	
SPP-Net [223]		Multi-region CNN [224]		
Overfeat [225]		HyperNet [226]		
DSSD [227]		IoU-Net [228]		
MDSSD [229]		Hybrid task cascade [230]		
DETR [231]				
EfficientDet [232]				

Table 3.1 **An overview of the categorization of CNN-based object detection methods for object detection on pinhole camera images.**

and can be used in any CNN-based classification model. On the other hand, Fast R-CNN and SPP-Net are not end-to-end trainable since they depend on a region proposal approach. Faster R-CNN [126] overcame this constraint by implementing the Region Proposal Network (RPN), which allowed end-to-end training. RPNs produce ROIs by regressing a series of reference boxes known as anchor boxes. R-FCN [221], which replaces fully connected layers with Fully Convolutional Network (FCN), improves the efficiency of Faster R-CNN [126].

Single-Stage Detectors

In contrast to the two-stage method, single-stage detectors skip the RoI extraction stage and go straight to anchor box classification and regression. Specifically, the RoI pooling phase is omitted, and object detection is accomplished with a single network. Using a multi-scale, sliding window technique, Overfeat [225] suggested a standardized framework to perform three tasks: classification, localization, and detection. "Overfeat," a feature extractor for vision applications, was introduced. You Only Look Once (YOLOv1) [38] is a single-stage detector that divides the input image into grids and predicts the BB directly using regression and classification. YOLO9000 (YOLOv2) [233] enhances efficiency by adding batch normalization and replacing YOLOv1's fully connected layers with anchor boxes for BB prediction. YOLOv3 [234], which is quicker and more reliable than previous models, employs Darknet-53 as its feature extraction backbone. YOLOv3 can detect small objects with multi-scale predictions, which was a significant limitation in previous versions.

The Single Shot Multibox Detector (SSD) [125] overlays dense anchor boxes on the input image and extracts feature maps at different scales. The anchor boxes are then classified and regressed to predict BB. ResNet101 [163] replaces the VGG network of SSD in DSSD [227]. It is then supplemented with a deconvolution module to merge feature maps from the beginning with the deconvolution layers. In terms of detecting small objects, it outperforms SSD. MDSSD [229] expands DSSD with fusion blocks to handle feature maps at various scales. During training, RetinaNet [216] implemented focal loss to resolve foreground and context class imbalance. It matches or exceeds the accuracy of cutting-edge two-stage detectors while operating at a faster rate. The design reuses RPN's 'anchors' and constructs an FCN with Feature Pyramid Network (FPN) [39] on top of the ResNet backbone.

SqueezeDet [220] focuses on small model size, speed, and accuracy, making it ideal for object detection in autonomous driving. It used the Yolov1 detection pipeline and created a ConvDet layer to generate region proposals with fewer parameters than YOLOv1. EfficientDet [232] is a one-stage detector that is powered by EfficientNet [235]. It proposed a weighted bi-directional FPN to fuse multi-scale features and compound scaling, which jointly scales up all networks' depth, width, and resolution. Detection Transformer (DETR) [231] is a recent work on the direct set prediction paradigm. To provide unique predictions, it employs a transformer-based encoder-decoder architecture and a bi-partite matching loss function. On large objects, DETR outperforms Faster-RCNN [126].

Multi-Stage Detectors

As the name suggests, detectors in this group use a series of CNNs at different levels to detect objects. Cascade R-CNN [210] is a multi-stage R-CNN [209] extension. The training data is sampled at each point, and the IoU threshold is increased. It uses iterative BB regression to advance the hypotheses. CRAFT [213] used a cascaded structure with RPN and two Fast R-CNNs [37] to introduce two functions, proposal generator and classifier, to improve the efficiency of proposal generation and detection. The CC-Net [217] is made up of several cascade stages. It has two stages of the cascade (i) early cascade and (ii) contextual cascade. Since shallow layers reject easy ROIs, hard samples are easily managed by later stages. Fast R-CNN is also used as the foundation for a Multipath Net [222] with a few modifications such as skip connections, foveal regions, and improved loss functions. Information flows through several paths in the network, allowing the classifier to operate at various scales. Fast R-CNN suggested a multi-region CNN model with an iterative BB regression process that switches between box scoring and coordinate refinement. HyperNet [226] is a multi-stage architecture that uses hyper feature maps to perform region proposal and object detection jointly. IoU-Net [228] achieves progressive BB regression using a standalone IoU predictor that can then be combined with any FPN-based CNN architecture for object detection.

Deformable Part based Detectors

Handling dynamic object deformation properties aids in improving detection efficiency. In [236], a deformation layer is proposed for pedestrian detection. Deformable Part Models (DPM) [211] is a single CNN that constructs a distance transform pooling layer to map an input image pyramid to a detection score pyramid. DeepIDNet [214] proposed a CNN for object detection based on a deformable part by designing a def-pooling layer to learn the geometric deformations of all instances of a part. DP-FCN [218] enhances it further by using a deformable part-based ROI pooling layer and deformation-aware localization. The model is fully convolutional end-to-end trainable. It focuses on discriminative elements. CNN's can learn dense spatial transformations with the aid of Deformable ConvNets [207], which can then be used in object detection tasks. It introduced two modules: (i) deformable convolution and (ii) deformable ROI pooling, which aid CNNs in modeling geometric transformations efficiently.

Keypoint based Detectors

In single-stage detectors, key points take the place of anchor boxes. To predict BBs, most one-stage detectors position dense anchor boxes over the image. RetinaNet [216] and DSSD [227], for example, require nearly 100k and 40k anchor boxes, respectively, thereby inducing hyperparameters. CornerNet [212] overcomes these limitations by detecting objects as paired key points representing the object's top left and bottom right corners. It employs an hourglass network as its backbone and employs corner pooling to locate corners. It uses associative embedding to group the key points. CenterNet [219] extends CornerNet by representing each object as a triplet – two corners and a central key point. It pioneered center pooling and cascade corner pooling to allow corners to recognize visual patterns of objects. ExtremeNet [215], on the other hand, detects BBs by looking at the topmost, leftmost, bottommost, rightmost, and the center of all objects.

3.2.2 Object Detection on Fisheye Cameras

Typically, automotive systems are equipped with a multi-camera network to cover the entire FoV around the vehicle [62]. The wide FoV of the fisheye image comes with the side effect of strong radial distortion. Objects at different angles from the optical axis look quite different, making the object detection task a larger challenge than for pinhole cameras (see Figure 2.14). All CNN-based detectors are trained with standard pinhole camera images that are free of any optical aberrations and closely resemble objects' primary form and appearance in the real world. There have not been many attempts to specialize existing CNNs to detect deformed images produced and affected by fisheye lenses. A common practice is to rectify distortions in the image using a 4th order polynomial [12] model or unified camera model [82]. However, undistortion comes with resampling distortion artifacts, especially at the periphery. In particular, the negative impact on computer vision due to the introduction of spurious frequency components is understood [237]. Other more minor impacts include a reduced FoV and a non-rectangular image due to invalid pixels. Although semantic segmentation is an easier solution on fisheye images, object detection annotation costs are much lower [27].

Agarwal *et al*. [238] provides a detailed survey of current object detection methods and their challenges. Presumably, fisheye camera object detection is a much more complex problem. The rectangular bounding box (BB) fails to be a good representation due to the massive distortion in the scene. The size of the standard BBs in a fisheye image is almost double the size of the object of interest inside it. Instance segmentation can help to obtain accurate object contours. However, it is a different task that is computationally complex and more expensive to annotate. It also typically needs a BB estimation step. There are few works on object detection for fisheye camera images or closely related omnidirectional cameras. One of the main issues is the lack of a useful datasets, particularly for autonomous driving scenarios. The recent fisheye object detection paper FisheyeDet [239] emphasizes the lack of a useful datasets, and they create a simulated fisheye camera dataset by applying distortions to the Pascal VOC dataset [151]. FisheyeDet makes use of a 4-sided polygon representation aided by distortion shape matching. SphereNet [240] and its variants [241, 242, 243] formulate CNNs on spherical surfaces. However, fisheye images do not follow spherical projection models, as seen by non-uniform distortion in horizontal and vertical directions. Deng *et al*. [244] pioneered the use of deep learning to detect multi-class objects in fisheye images. Yang *et al*. [245] compared the results of various detection algorithms that take equirectangular projection (ERP) images directly as inputs, demonstrating that the network produces only a certain accuracy without projecting ERP images into conventional 2D images.

Compared to the state-of-the-art approaches, in this thesis, we explore different object representations for fisheye object detection and design novel representations for fisheye images. We also release a dataset of 10,000 images with annotations for all the object representations. We perform an empirical study of our baseline, which can output different representations.

3.3 Semantic Segmentation

Semantic segmentation is the task of assigning dense semantic labels to images. It is of paramount significance in computer vision and has applications in autonomous driving, augmented reality, and human-computer interaction.

3.3.1 Semantic Segmentation on Pinhole Cameras

A multiscale convolutional network by Farabet *et al.* [246] is fused with a segmentation framework in parallel (either superpixel or CRF-based). Because of a CRF block, computational efficiency is reduced. Pinheiro *et al.* [247] built a recurrent architecture by using multiple instances of a CNN, each of which is fed with previous label predictions (obtained from the previous instance). There is a significant computational load when multiple instances (3 in their best-performing experiments) are fed. A standard milestone approach for semantic segmentation is the introduction of fully convolutional neural networks by Long *et al.* [40]. The network architecture is a fully convolutional encoder structure (no fully connected layers) with skip connections at the final decision layer that fuse multiscale activations. Due to the lack of fully connected layers or a refinement block, it is comparably fast. DeepLabv1 [248] a CNN with dilated convolutions is followed by a fully-connected (*i.e.*, Dense) CRF. Near-real-time performance is achieved through fast and optimized computation [148].

In parallel, layers of a pyramidal input are fed to separate FCNs for different scales by Eigen *et al.* [51]. These multiscale FCNs are also linked in series to provide pixel-by-pixel category, depth, and normal output simultaneously—lower computational efficiency as a result of progressive processing of a sequence of different scales. The UNet [249] architecture uses an encoder-decoder (ED) structure with skip connections, connecting the same ED and final input-sized classification layer levels. Due to the lack of fully connected layers or a refinement block, the computation load is efficient. The SegNet [42] structure is similar to UNet, but with skip connections that only transmit pooling indices (unlike U-Net, where skip connections concatenate the same level activations). Due to the lack of fully connected layers or a refinement block, the computation load is efficient. The DeconvNet [41] structure is comprised of an ED (referred to as 'the Conv./Deconv. Network') with no skip connections. The network's (convolutional) encoder component is transferred from the VGG-VD-16L [250]. Due to the lack of fully connected layers or a refinement block, the computation load is efficient. To perform pixel-wise labeling MSCG [251], designed multiscale context aggregation using only a rectangular prism of dilated convolutional layers without pooling or subsampling layers. Due to the lack of fully connected layers or a refinement block, the computation load is reduced. Zhen *et al.* [252] formulated a CRF-as-RNN, *i.e.*, an FCN is followed by a CRF-as-RNN layer, which translates an iterative CRF algorithm into an RNN. Computational efficiency is limited due to the RNN block [148].

FeatMap-Net [253] constitutes layers of a pyramidal input fed to parallel multiscale feature maps (*i.e.*, CNNS), which were then fused in an upsample/concatenation (*i.e.*, pyramid pooling) layer to provide the final feature map to a Dense CRF Layer. A well-thought-out but overburdened architecture results in moderate computational efficiency. Graph Long Short-Term Memory (LSTM) [254] is a LSTM generalization from sequential data to general graph-structured data for semantic segmentation, primarily of people. DAG-RNN [255] is a CNN+RNN network with a DAG structure that models long-term semantic dependencies among image units. The computational efficiency is significantly limited due to chain structured, sequential processing of pixels with a recurrent model. DeepLabv1 has been improved by Chen *et al.* [256], with the addition of a 'dilated (atrous) spatial pyramid pooling (ASPP) layer. PSP-Net [257] consists of a CNN followed by a pyramid pooling layer similar to [223], but it lacks a fully connected decision layer. As a result, computational performance is closer to FCN [40]. DeepLabv2 has been improved, with ASPP layer hyperparameters optimized and non-dense CRF layer, for faster operation by [258]. Luo *et al.* [259] introduces dual learning for semantic image segmentation where one network predicts label maps/tags, while another uses these predictions to perform semantic segmentation. ResNet101 [163] which is a larger backbone is used in both networks for preliminary feature extraction [148].

GCN [260] uses large kernels to fuse high- and low-level features in a multiscale manner, powered by an initial ResNet-based [163] encoder. Stacked deconvolutional network [261] is a UNET architecture made up of multiple shallow deconvolutional networks, known as SDN units, that are stacked one on top of each other to integrate contextual information and to ensure fine recovery of localized information. Discriminative Feature Network [262] is made up of two sub-networks: Smooth Net (SN) and Border Net (BN). SN makes use of an attention module to handle global context, whereas BN makes use of a refinement block to handle borders. Due to an attention block, computational efficiency is limited. Multi-Scale Context Intertwining [263] connects LSTM chains to aggregate features from different scales. Due to multiple RNN blocks, the computational efficiency is limited (*i.e.*, LSTMs). DeepLab.v3+ [264] is an improved version of DeepLab.v3 that employs a special encoder-decoder structure with dilated convolutions (rather than using Dense CRF for faster operation). Hierarchical Parsing Net [265] is a convolutional 'Appearance Feature Encoder,' while a 'Contextual Feature Encoder' made up of LSTMs generates superpixel features that are fed into a Softmax-based classification layer. Due to the use of multiple LSTMs, computational efficiency is limited. EncNet [266] is a fully connected structure fed by dense feature maps (obtained from ResNet) and followed by a convolutional prediction layer to extract context. Fully connected layers limit computational performance within their "Context Encoding Module" [148].

PSANet [267] is a convolutional point-wise spatial attention (PSA) module connected to a pretrained convolutional encoder, allowing pixels to be interconnected via a self-adaptively learned attention map to provide global context. When compared to fully convolutional architectures (*e.g.*, FCN), the addition of a PSA module reduces computational efficiency. EMANet152 [268] is made up of a novel attention module that converts input feature maps to output feature maps, providing global context. When compared to other attention governing architectures, it is computationally more efficient (*e.g.*, PSANet). Kernel-Sharing Atrous Convolution [269] enables branches from different receptive fields to use the same kernel, allowing for more accessible branch communication and feature augmentation within the network. In Co-occurrent features (CFNet) [270] a fine-grained spatial invariant representation is learned. The CFNet is constructed using a distribution of co-occurrent features for a given target in an image.

In this task, the right architectural design choice and multi-scale context modules are vital for performance. The former depend upon spatial pyramid pooling [223, 257] and atrous convolutions [256, 258, 264]. For the latter, the contemporary advancements in the design [271, 272] have improved the popular backbones [163, 250, 273]. Despite their popularity, they require huge computational requirements and are not real-time capable for inference. The most popular design choice has been the encoder-decoder architecture [42, 249]. To obtain state-of-the-art accuracy, recent approaches have incorporated Auto Machine Learning (AutoML) [274, 275]. ENet [276] is a compact real-time capable efficient network which follows an encoder-decoder architecture design. Bisenet [277] is a two-path network designed to achieve fast inferences while obtaining high-resolution details. DABNet [278] combines the depthwise separable filters and atrous-convolutions to obtain a decent trade-off between accuracy and efficiency. DfaNet [279] employs cascaded sub-stages to refine the segmentation predictions. To maximize larger networks' performance during inference, FCHardNet [280] leverages on a new harmonic densely connected pattern.

3.3.2 Semantic Segmentation on Fisheye Cameras

Most of the popular semantic segmentation studies listed above are based on images taken by pinhole cameras. However, the urban traffic conditions are so complicated that more knowledge of the surroundings is needed, while the pinhole camera only has a narrow FoV. If a vehicle or pedestrian suddenly arrives from a blind spot, the safe operation of autonomous driving is hard to ensure. One of the methods is to enhance the significance of the information obtained for holistic scene comprehension. An Overlapping Pyramid Pooling module (OPP-Net) was presented by Deng *et al.* [281] by employing several focal lengths to simulate different fisheye images with their corresponding annotations. In order to achieve real-time semantic segmentation, Saez *et al.* [282] introduced an adaptation of Efficient Residual Factorized Network (ERFNet) [283] to fisheye road images. The tests were performed on authentic fisheye images, but only qualitative results were revealed. Deng *et al.* [208] used the identical approach to obtain road scene semantic segmentation of fisheye surround-view cameras using restricted deformable convolution. These models were trained on Cityscapes [284] and SYNTHIA [285] datasets and tested on authentic fisheye images. However, to perform the segmentation directly on the fisheye images, we would require a large-scale finely-annotated fisheye image dataset to train the network. *Henceforth, as a contribution to the research community, we release the WoodScape [12] dataset.* Figure 2.16 depicts the semantic segmentation task on the WoodScape dataset.

Panoramic Images

Xu *et al.* [286] created a dataset of panoramic images by stitching images taken from different directions using synthetic images captured from SYNTHIA. The authors demonstrate that panoramic images improve segmentation results using these images. Yang *et al.* [287] proposes a panoramic angular semantic segmentation framework by creating a data augmentation method by adding distortion to perspective images for the training set. After unfolding and partitioning the panoramic images, normal CNNs were used.

Equirectangular Images

Because of the simple transformation from spherical coordinates to planar coordinates, equirectangular representation is the most popular projection for 360° images. Classical CNNs designed for perspective images can be applied to equirectangular data. In polar regions, however, spherical input suffers from distortion. To address this issue, various approaches were proposed. SalNet360 was proposed by Monroy *et al.* [288], in which omnidirectional images were mapped to cube map by six faces projection and trained using standard CNNs to predict visual attention. However, artifacts are produced when the cube map faces are recombined to create an omnidirectional image. Lai *et al.* [289] converted panoramic videos to normal perspective images using semantic segmentation of equirectangular images. However, because highly accurate semantic labels was not required for this task, a frame-based fully convolutional network FCN [40] was used in this work. For the same task, they proposed the kernel transformer network, which efficiently transfers convolution kernels from perspective images to the equirectangular projection of 360° images. Tateno *et al.* [290] proposed a learning method for equirectangular images that uses a distortion-aware deformable convolution filter to estimate depth from a single image, and this method was also demonstrated on 360° semantic segmentation.

Spherical Representations

Due to distortions caused by the equirectangular representation, the most recent work on this topic has focused on the spherical presentation. Cohen *et al.* [291] created spherical convolutions by substituting sphere rotations for plane translations. Other works used the icosahedral spherical approximation, which is the most precise sphere discretization. To represent the discretization of the sphere, a spherical mesh is generated by dividing each face of a regular icosahedron into four equal triangles. In the case of triangle faces, Lee *et al.* [292] proposed an orientation-dependent kernel method, which was demonstrated through classification, detection, and semantic segmentation. Zhang *et al.* [293] proposed an orientation-aware CNN framework for semantic segmentation on omnidirectional images using icosahedron spheres. UGSCNN was proposed by Jiang *et al.* [243] to train spherical data mapped to an icosahedron mesh by replacing conventional convolution kernels with linear combinations of learnable weighted operators.

3.4 Motion Segmentation

The classical approach to detecting moving objects is based on the scene's geometrical understanding, where the ego-vehicle motion and the displacement vectors of the pixels between two frames are known. Classical methods for moving object detection based on the geometrical understanding of the scene have been suggested, such as [144], which was used to estimate object motion masks. To model the background motion in terms of homography, Wehrwein *et al.* [294] introduced assumptions about the camera motion model. Due to the errors caused by the restricted assumptions, such as camera translations, this method cannot be used in autonomous driving applications. Classical approaches have a lower performance than deep learning methods and a high level of difficulty due to the complicated pipelines used. The method of Menze *et al.* [144], for example, has a running time of 50 minutes per frame, making it unsuitable for use in a real-time application such as autonomous driving [295].

Jain *et al.* [296] introduced a method for generic foreground segmentation that takes advantage of optical flow. This work is intended for generic object segmentation and does not concentrate on object classifications as *Moving* or *Static*. Drayer *et al.* [297] proposed an R-CNN detection-based video segmentation algorithm. Due to its sophistication, the technique is not feasible for autonomous driving applications, where it takes 8 seconds to infer an image. Arguably the most famous constraint used in motion detection is the epipolar constraint [298, 299], which can be combined with additional geometrical constraints to detect multiple types of motion [300]. However, even if the moving objects' geometry is well known, their detection still presents challenges caused by intrinsic geometrical limitations. Many methods for segmenting moving objects from stationary camera images have been suggested in [301, 302]. However, they cannot be directly applied to moving camera images because movement causes a dual motion appearance consisting of background motion and object motion. Methods that detect motion from freely moving cameras, in general, divide the image into coherent regions with homogeneous motion. The image is divided into the background and moving clusters during this process. These approaches can be divided into two types: optical flow-based and tracking-based approaches. Optical flow-based techniques [303, 304] determine whether a region's motion speed and direction are compatible with its radially surrounding pattern. Tracking-based methods, on the other hand [45, 46, 297, 305], tend to track and localize target points in successive frames. Object tracking produces movement trajectories, and by estimating the camera's ego-motion, objects can be separated from the background motion. Large processing pipelines are common in these methods, resulting in long computation times and coarse segmentations.

Chen *et al.* [306] suggests a method for detecting object-level motion from a moving camera using two consecutive image frames and outputs 2D BBs. They create a robust context-aware motion descriptor that considers movement speed and object direction and combines it with an object classifier. The descriptor calculates the discrepancy between local optical flow histograms of objects and their surroundings, yielding a state of motion measurement. Dinesh *et al.* [307] propose a method for generating motion likelihoods based on depth and optical flow estimations while incorporating semantic and geometric constraints within a dense CRF. Fan *et al.* [308] suggested a multistep framework in which first sparse image features in two consecutive stereo image pairs are extracted and matched. RANSAC is then used to classify the matched feature points as inliers caused by the camera and outliers caused by moving objects. The outliers are then clustered in a U-disparity chart, which provides object motion information. Finally, the motion information is combined with the semantic segmentation given by an FCN using a dense CRF. Long run times, ranging from a few seconds to minutes, are a major drawback of these methods, rendering them unsuitable for applications that involve near-real-time efficiency, such as autonomous driving. In the search to overcome the limitations of the classical approach, there has been good work in using CNN to solve the moving object detection problem, such as MODNet [28], FisheyeMODNet [21], MPNet [309], OmegaNet [304], SMSnet [49] and Ranjan *et al.* [303]. Siam *et al.* [28, 310] investigated motion segmentation using deep network architectures, but these networks depend solely on camera RGB images, which can fail in low-light conditions.

Given the use of fisheye cameras in surround-view systems, it is of utmost importance for research to explore this direction and provide a CNN architecture for moving object detection on fisheye images. One of the main challenges of detecting moving objects with a CNN is to make it scene agnostic so that the detection is based only on motion cues & not on appearance cues. Figure 2.17 depicts the motion segmentation task on the WoodScape dataset.

3.5 Soiling Segmentation

There is very little work on the related but distinct lens soiling problem. There are two types of soiled areas: opaque (mud, dust, snow) and translucent (water). Transparent soiling, in particular, can be challenging to detect due to the background's partial visibility. The two problems are similar in how they degrade image quality and can severely affect visual perception performance. However, there are substantial differences. The first significant difference is that soiling on the lens can be removed by a camera cleaning system that either sprays water or uses a more sophisticated ultrasonic hardware [43]. Secondly, there is temporal consistency for soiling where mud or water droplets remain static typically or sometimes have low-frequency dynamics of moving water droplets compared to higher variability in adverse weather scenes. Thus this temporal structure can be exploited further for soiling scenarios. Finally, soiling can cause more severe degradation as opaque mud soiling can completely block the camera. Porav *et al.* [311] discussed transparent soiling in a recent study in which a stereo camera was used in combination with a dripping water supply to simulate raindrops on the camera lens. The authors also suggest a CNN-based de-raining algorithm. Sakaridis *et al.* [312] suggested a robust semantic segmentation algorithm that can handle foggy scenes. However, dealing with opaque soiling in this manner would be difficult. To boost the image quality, the third solution is to run a separate image restoration algorithm. De-raining [311, 313, 314, 315, 316, 317] is a recent example of restoration algorithms in automotive scenarios. This will primarily help with partial soiling. Restoration algorithms may be single-image or video-based. The latter is more computationally costly, but it can benefit from the visibility of soiling occluded regions over time [73].

In this thesis, we focus on the generic soiling detection task. Even disregarding camera cleaning, soiling detection is still needed to increase vision algorithms' uncertainty in the degraded areas. A more formal introduction to the soling detection and categorization is provided in [43]. The problem is formalized as a multi-label classification task and discusses soiling detection applications, including camera cleaning. The authors present a proof of concept idea on how GANs [165] could be applied for dealing with the insufficient data problem in terms of an advanced data augmentation. The authors also outline another potential usage of GANs in the AD area. Uřičář *et al.* [73] provided a desoiling dataset benchmark. SoildNet was explored in [318] for embedded platform deployment.

3.6 Multi-Task Learning

Multi-task learning (MTL) is carried out by learning commonly shared representations from multi-task supervisory signals. Many dense prediction tasks, *i.e.* tasks that generate pixel-level predictions, have seen substantial performance improvements since the introduction of deep learning. Typically, these tasks are learned one at a time, with each task requiring its own neural network to be trained. However, by jointly tackling multiple tasks via a learned shared representation, recent MTL techniques have shown promising results regarding performance, computational complexity, and memory footprint.

MTL [130] seeks to enhance generalization by incorporating domain-specific knowledge found in similar task training signals. MTL refers to the design of networks capable of learning mutual representations from multi-task supervisory signals. Compared to the single-task example, where each network solves only one task, multi-task networks provide several benefits. First, the resulting memory footprint is significantly reduced due to their intrinsic layer sharing. Second, since they explicitly avoid calculating the features in the shared layers multiple times for each task, they demonstrate faster inference speeds. Most significantly, whether the related tasks exchange complementary knowledge or function as a regularizer for one another, they have the potential to improve results [319].

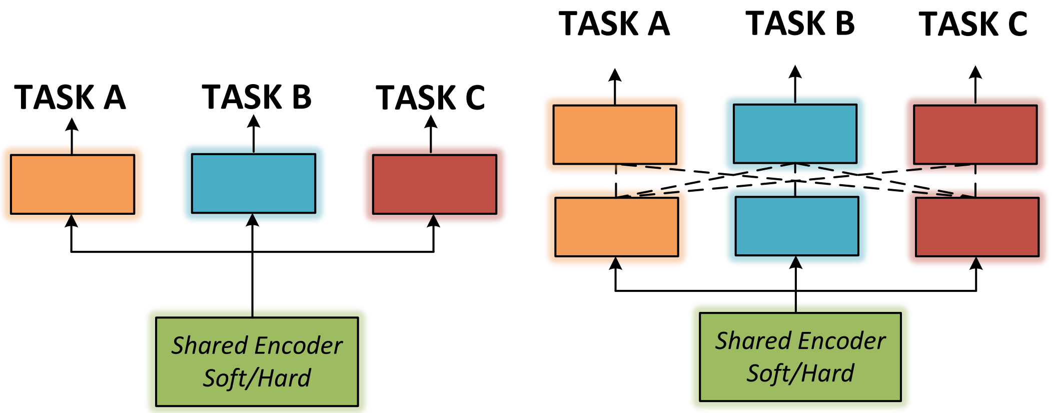
Non-Deep Learning-Based Methods

Before the deep learning era, MTL works attempted to model the common information among tasks to improve generalization performance through joint task learning. To do so, they imposed constraints on the task parameter space, such as: task parameters should be close to each other in terms of some distance metric [320, 321, 322, 323], share a common probabilistic prior [324, 325, 326, 327, 328], or reside in a low-dimensional subspace [329, 330, 331] or manifold [332]. These assumptions work well when all tasks are related [320, 329, 333, 334] and regularly scheduled, but they can degrade performance if the information is shared between unrelated tasks. The latter is a well-known MTL issue known as "*negative transfer*." To address this issue, some of these studies chose to group tasks based on prior assumptions about their similarity or relatedness [319].

Distilling Task Predictions in Deep Learning

All of the works in Section 2.3.2: soft vs. hard have one thing in common: they *directly* predict all task outputs from the same input in a single processing cycle. On the other hand, several recent studies used a multi-task network to make initial task predictions. They then used features from these initial predictions to improve each task output in a one-off or recursive manner. PAD-Net [335] proposed using spatial attention to distill information from initial task predictions of other tasks before adding it as a residual to the task of interest. JTRL [336] chose to predict each task by using information from previous predictions to refine the features of another task at each iteration. PAP-Net [337] built on this concept by employing a recursive procedure to propagate similar cross-talk and task-specific patterns discovered in the initial task predictions. They did so by using the affinity matrices of the initial predictions rather than the features themselves, as was the case previously [335, 336]. By separating inter-and intra-task patterns from each other, Zhou *et al.* [338] refined the use of pixel affinities to distill the information. To explicitly model the unique task interactions that occur at each scale, MTI-Net [339] used a multi-scale multi-modal distillation procedure [319].

Multi-task networks have traditionally been classified as using soft or hard parameter sharing techniques, as explained in Section 2.3.2. However, several recent works drew



(a) Encoder-focused model

(b) Decoder-focused model

Figure 3.1 **Illustration of the encoder and decoder-focused models depending on where the task synergies take place.**

inspiration from both groups of works to collaboratively solve multiple pixel-level tasks. As a result, whether the soft vs. hard parameter sharing paradigm should still be used as the primary framework for classifying MTL architectures is debatable. An alternative taxonomy distinguishes between various architectures based on the interactions between tasks, *i.e.*, network locations where information and features are shared or exchanged between tasks.

MTL is typically divided into two sections based on the proposed criterion: shared parameters and task-specific parameters. We allow the shared parameters to learn representing the commonalities between many tasks, while task-specific parameters are learned to perform independent task-specific processing. Shared parameters are commonly dubbed as *encoders* (see Figure 3.1) wherein they carry out the key feature extraction. The task-specific parameters are called *decoders* (see Figure 3.1) as they decode the vital information from the encoders. This method presents benefits such as enhanced data efficiency, reduced overfitting through shared representations, and fast learning by leveraging auxiliary information [340]. It is an efficient design pattern commonly used where most of the computation can be shared across all tasks [198, 319]. Besides, learning features for multiple tasks can act as a regularizer, to improve generalization.

3.6.1 Encoder-focused Architectures

The information is shared only in the encoder by employing either hard or soft-parameter sharing by the encoder-focused architectures before decoding each task with an independent task-specific head. Conversely, the decoder-focused architectures further exchange information during the decoding stage. Most recent works [132, 133, 134, 135, 136] followed an ad-hoc approach by sharing an off-the-shelf backbone model in union with small task-specific heads. The activations were shared amongst all single-task models in the encoder by, *e.g.*, cross-stitch networks proposed in [141]. Yuan *et al.* [341] developed neural discriminative dimensionality reduction CNNs, a similar architecture as cross-stitch networks. It used a dimensionality reduction mechanism rather than employing a linear combination to fuse all single-task networks' activations. In combination with task-specific attention modules in the encoder, Liu *et al*. [342] introduced multi-task attention networks which employed a shared backbone model.

3.6.2 Decoder-focused Architectures

Regarding decoder-focused architectures, PAD-Net [335] was one of the first approaches. It performs multi-modal distillation through a spatial attention mechanism. Zhang *et al*. [337] introduced Pattern-Affinitive Propagation Networks (PAP-Net), which incorporated a similar architecture as PAD-Net. PAP-Net involves leveraging pixel affinities in order to perform multi-modal distillation. Joint Task-Recursive Learning (JTRL) [336] recursively estimates two tasks at frequently higher scales to slowly improve the decisions based on past states. The architecture is similar to PAD-Net and PAP-Net; a multi-modal distillation mechanism is employed to couple information from earlier task predictions, through which later estimates are improved. Conversely, the JTRL model estimates two tasks sequentially rather than parallel and in an intertwined manner. In the above-listed decoder-focused works, multi-modal distillation was performed at a fixed scale, *i.e.*, the backbone's last layer features. Nevertheless, Multi-Scale Task Interaction Networks (MTI-Net)[339] indicated that this is somewhat a strict assumption. MTI-Net explicitly took into account task interactions at multiple scales.

3.6.3 Other Approaches

To enable synergies in the decoding stage, Multilinear relationship networks [343] employed tensor normal priors to the parameter set of the task-specific heads. Compared to [141, 341] where layers are aligned and shared according to the standard parallel ordering scheme, Elliot *et al*. [344] proposed soft layer ordering a flexible sharing scheme across tasks and network depths. Yang *et al*. [345] generalized matrix factorization methods to MTL to learn cross-task sharing structures in every layer of the model. Clemens *et al*. [346] proposed routing networks as a principled way to determine the connectivity of a model's function blocks through routing. By learning binary masks, Piggyback [347] demonstrated how to adapt a single, fixed neural network to a multi-task network. Huang *et al*. [348] presented an approach rooted in neural architecture search for the automated creation of a tree-based multi-attribute learning model. Bragman *et al*. [349] re-formulated the convolution kernels in each layer of the network with stochastic filter groups to support either shared or task-specific behavior. In a similar style, Newell *et al*. [350] exhibited feature partitioning approaches to designate the convolution kernels in each layer of the model into different tasks. Overall, these works have a different scope within MTL, *e.g.*, automate the network architecture design. Furthermore, they focus on solving multiple (binary) classification tasks rather than numerous dense prediction tasks. As a result, they fall outside the scope of this thesis.

Maninis *et al*. [351] presented *Attentive Single-Tasking of Multiple Tasks* to take a single-tasking route for the MTL problem. *i.e.* inside an MTL framework, they made separate forward passes, one for each task, that initiate shared responses among all tasks, complemented by residual responses that are task-specific. Moreover, adversarial training on the gradient level was applied to be statistically identical across tasks to overcome the negative transfer problem. A benefit of this method is that shared and task-specific knowledge inside the model can be easily disentangled. On the negative side, the tasks can not be estimated altogether, but only sequentially, which significantly increases the inference speed and somehow defies the purpose of MTL.

Recently, Mao *et al*. [352] illustrated that multi-task learning improves adversarial robustness, which is critical for safety applications. In the automotive multi-task setting, MultiNet [136] was one of the first to demonstrate a three task network on KITTI, and most further works have primarily worked on a three task setting. In contrast to letting a network predict a single task, it is also possible to train a network to predict several tasks at once (see Figure 2.21). It has shown to improve tasks such as, *e.g.*, semantic segmentation, [198, 199, 353, 354, 355], domain adaptation [356, 357, 358], instance segmentation: [359], and depth estimation [51, 132, 360, 361].

3.6.4 Previous MTL based Semantically-Guided Distance Estimation

As the distance predictions are still imperfect due to the monocular cues such as occlusion, blur, haze, and different lighting conditions and the dynamic objects during the self-supervised optimizations between consecutive frames. Many approaches consider different scene understanding modalities, such as segmentation [7, 303, 362] or optical flow [363, 364] within multi-task learning to guide and improve the distance estimation. As optical flow is usually also predicted in a self-supervised fashion [365], it is, therefore, subject to similar limitations as the self-supervised distance estimation, which is why we focus on the joint learning of self-supervised distance estimation and semantic segmentation.

To improve the distance estimation task in MTL, many recent approaches aim to integrate optical flow into the self-supervised distance estimation training. This additional task can also be trained in a self-supervised fashion [365, 366]. In these approaches, both tasks are predicted simultaneously. Then losses are applied to enforce cross-task consistency [363, 367, 368, 369], to enforce known geometric constraints [303, 364], or to induce a modified reconstruction of the warped image [364, 370]. Although the typical approach is to compensate using optical flow, in this thesis we propose an alternative method to use semantic/motion segmentation instead for two reasons. (i) Semantic segmentation is a mature and common task in autonomous driving, which can be leveraged. (ii) Motion segmentation is an easier problem to solve as it falls under segmentation tasks (similar to semantic segmentation). It can be learned in a supervised fashion compared to optical flow, which is computationally more complex and harder to validate because of difficulties in obtaining ground truth.

Depth has no mathematical relationship with semantics. Several works [7, 303, 362], however, follow this direction on the basis of the concept that semantic can guide depth estimation by offering specific cues. Sky, for example, should be placed far away and naturally with very high depth values. A shift in pixel mark will most likely signify an object's boundary, resulting in a noticeable change in depth. Several recent approaches also used semantic or instance segmentation techniques to identify moving objects and handle them accordingly inside the photometric loss [7, 157, 188, 189, 362]. To this end, the segmentation masks are either given as an additional input to the network [7, 362] or used to predict poses for each object separately between two consecutive frames [157, 188, 189] and apply a separate rigid transformation for each object. Avoiding an unfavorable two-step (pre)training procedure, other approaches in [371, 372, 373, 374] train both tasks in one multi-task network simultaneously, improving the performance by cross-task guidance between these two facets of scene

Method	*Balancing Magnitudes*	*Balance Learning*	*Prioritize*	*Gradients Required*	*No Extra Tuning*	*Motivation*
Uncertainty [132]	✓	✗	Low Noise	✗	✓	Homoscedastic uncertainty
GradNorm [133]	✓	✓		✓	✓	Balance learning & magnitudes
DWA [342]	✗	✓		✗	✗	Balance learning
DTP [375]	✗	✗	Difficult	✗	✗	Prioritize difficult tasks

Table 3.2 **Ablation of different task balancing techniques [319].** Firstly, we contemplate if an approach balances the loss magnitudes (*Balance Magnitudes*) and/or the speed at which tasks are learned (*Balance Learning*). Also, we attest what tasks are prioritized during the training stage (*Prioritize*) and followed by if the approach needs access to the task-specific gradients (*Gradients Required*). Finally, we consider if the suggested approach needs additional tuning, *e.g.* manually determining the weights, and KPIs (*No Extra Tuning*).

understanding. Moreover, the segmentation masks can be projected between frames to enforce semantic consistency [371, 373], or the edges can be enforced to appear in similar regions in both predictions [371, 374]. In this thesis, we propose to use this warping to discover frames with moving objects and learn their depth from these frames by applying a simple semantic masking technique.

3.6.5 Optimization in MTL

In the previous section, we reviewed literature modeling MTL architectures that can learn multiple tasks together. However, a vital challenge in MTL arises from the optimization method itself. In particular, we need to thoughtfully weigh the joint learning of all tasks to circumvent a state where one or more tasks have an imperative influence on the network weights. This section considers various techniques that have studied this *task weighing* problem. While initial works did weigh losses [51] or gradients [376] by an empirical factor, current approaches can estimate this scale factor automatically [132, 133].

Homoscedastic uncertainty to was incorporated to weigh the single-task losses by Kendall *et al*. [132]. The homoscedastic uncertainty or *task-dependent uncertainty* is not an output of the network rather a quantity that remains constant for different input samples of the corresponding task. Chen *et al*. [133] proposed **Gradient normalization** (GradNorm) to command the training of multi-task networks by stimulating the *task-specific gradients* to be of similar magnitude. This encourages the network to learn all the assigned tasks at an equal pace. Similar to GradNorm, Liu *et al*. [342] proposed **Dynamic Weight Averaging** (DWA) to weigh the pace at which different tasks are learned. Compared to GradNorm, DWA requires only the task-specific loss values during training and eliminates the need to obtain the task-specific gradients separately by performing backward passes each time. The task weighting techniques [132, 133, 342] chose to optimize the task-specific weights as part of a Gaussian likelihood objective. In contrast, **Dynamic Task Prioritization** (DTP) [375] chose to prioritize the learning of ’challenging’ tasks by designating them a higher task-specific weight. The motive is that the network should pay more effort to learn the ’challenging’ tasks. This approach is entirely opposite to Kendall’s uncertainty weighting, where a higher weight is assigned to the ’easy’ tasks. Kendall’s approach suits better when tasks have noisy labeled data, while DTP performs better when we have the availability to clean ground-truth labels. Finally, the ablation of all the listed approaches is shown in Table 3.2.

3.7 Adversarial Attacks

For the first time, Szegedy *et al.* [377] demonstrated box-constrained L-BFGS, *i.e.*, small perturbations in the images so that perturbed imagery can fool deep learning models. Fast gradient sign method (FGSM) [378] is an example of a simple yet effective attack for generating adversarial instances. FGSM aims to fool the image classification by adding a small vector obtained by taking the sign of the gradient of the loss function. Moreover, it was shown that robust 3D adversarial objects could fool deep network classifiers in the physical world [379], despite the combination of viewpoint shifts, camera noise, and other natural transformations. The one-step methods cause images to be perturbed by taking a single significant step towards the classifier's loss (*i.e.*, one-step gradient descent). They are iteratively taking multiple small steps while adjusting the direction after each step is an intuitive extension of this idea. This is precisely what the Basic Iterative Method [380] does. Kurakin *et al.* [380] also extended BIM to the Iterative Least-likely Class Method (ILCM).

Papernot *et al.* [381] also devised a jacobian-based saliency map attack (JSMA), an adversarial attack by limiting the perturbation's L_0-norm. Physically, the goal is to change only a few pixels in the image rather than disrupt the entire image to fool the classifier. When only one pixel in an image is changed to fool the classifier, this is an extreme case of an adversarial attack. Surprisingly, Su *et al.* [382] claimed that by changing just one pixel per image, they were able to fool three different network models on 70.97% of the images tested. Carlini and Wagner (C&W) [383] proposed a set of three adversarial attacks in the aftermath of defensive distillation against adversarial perturbations [384]. These attacks make perturbations almost imperceptible by limiting their L_0, L_2, and L_∞ norms. It is demonstrated that defensive distillation for the targeted networks almost completely fails against these attacks. Moosavi-Dezfooli *et al.* [385] proposed to iteratively compute a minimal norm adversarial perturbation for a given image. DeepFool, their algorithm, starts with a clean image assumed to be in a region bounded by the classifier's decision boundaries. Whereas methods like FGSM [378], ILCM [380], DeepFool [385], and others compute perturbations to fool a network on a single image, Moosavi-Dezfooli *et al.* [386] compute 'universal' adversarial perturbations that can fool a network on 'any' image with high probability [387].

Sarkar *et al.* [388] proposed two black-box attack algorithms for targeted fooling of deep neural networks: UPSET: Universal Perturbations for Steering to Exact Targets and ANGRI: Antagonistic Network for Generating Rogue Images. Cisse *et al.* [389] proposed 'Houdini,' a method for deceiving gradient-based learning machines by generating adversarial examples that can be tailored to task losses. Houdini has also been demonstrated to be capable of successfully attacking a popular deep Automatic Speech Recognition system [390]. Feed-forward neural networks were trained by Baluja and Fischer [391] to generate adversarial examples against other targeted networks or sets of networks. Adversarial Transformation Networks were the name given to the trained models (ATNs). Hayex and Danezis [392] also used an ATN to learn adversarial examples for black-box attacks in the same direction [387]. The summary of characteristics of various attacking methods are shown in Table 3.3.

Attacks on Geometric and Semantic Tasks

Inspired by Moosavi-Dezfooli *et al.* [386], Metzen *et al.* [393] demonstrated the existence of image-agnostic quasi-imperceptible perturbations that can fool a deep neural

Method	*Black/ White box*	*Image/ Universal*	*Perturbation Norm*	*Learning*	*Strength*
L-BFGS [377]	*wb_target*	Image	ℓ_∞	One shot	∗ ∗ ∗
FGSM [378]	*wb_target*	Image	ℓ_∞	One shot	∗ ∗ ∗
BIM & ILCM [380]	*wb_untarget*	Image	ℓ_∞	Iterative	∗ ∗ ∗∗
JSMA [381]	*bb_untarget*	Image	ℓ_0	Iterative	∗ ∗ ∗
C&W attacks [383]	*wb_untarget*	Image	ℓ_2, ℓ_∞	Iterative	∗ ∗ ∗∗
DeepFool [385]	*bb_target*	Universal	ℓ_2, ℓ_∞	Iterative	∗ ∗ ∗ ∗ ∗
Universal perturbations [386]	*bb_target*	Image	ℓ_∞	Iterative	∗ ∗ ∗∗
UPSET [388]	*bb_target*	Image	ℓ_2, ℓ_∞	Iterative	∗ ∗ ∗∗
ANGRI [388]	*wb_target*	Image	ℓ_∞	Iterative	∗ ∗ ∗∗

Table 3.3 **A summary of the characteristics of various attacking methods:** The 'perturbation norm' denotes the perturbations' restricted p-norm in order to make them imperceptible. The strength (higher for more asterisks) is based on the review of the literature [387].

network into significantly corrupting the predicted image segmentation. Furthermore, they demonstrated that it is possible to compute noise vectors that can remove a specific class from the segmented classes while leaving most of the image segmentation intact (*e.g.*, removing pedestrians from road scenes). Even though the "space of adversarial perturbations for semantic image segmentation is presumably smaller than image classification," the perturbations have been shown to generalize well for unseen validation images with high probability. Arnab *et al*. [394] investigated FGSM [378] based adversarial attacks for semantic segmentation and discovered that several findings of these attacks for classification do not directly transfer to the segmentation task. Xie *et al*. [395] computed adversarial examples for semantic segmentation and object detection, observing that these tasks can be formulated as classifying multiple targets in an image - the target in segmentation is a pixel or a receptive field, and the target in detection is an object proposal. According to this viewpoint, their method, 'Dense Adversary Generation,' optimizes a loss function over a set of pixels/proposals to generate adversarial examples. The generated examples are tested for their ability to fool various deep learning-based segmentation and detection approaches. Their experimental results show that the generated perturbations not only fool the targeted networks but also generalize well across different network models [387]. In addition, research on adversarial attacks for monocular depth estimation began in 2019. Van *et al*. [396] developed some exceptional cases to investigate the internal mechanism of how networks perceive depth from images by constructing prominent fake images that can be identified at first glance. Yamanaka *et al*. [397] demonstrated that the target area's depth is incorrectly estimated by overwriting the local area of the input image. Mopuri *et al*. [398] proposed a data-free method for creating universal adversarial examples for a specific CNN. Although their method effectively uses depth estimation and semantic segmentation, it is limited to a single network. It cannot produce universal examples that attack both tasks from different structures at the same time. Hu *et al*. [399] investigated white-box adversarial attacks (I-FGSM) on depth estimation in an indoor setting and proposed a saliency map defense.

Fooling surveillance cameras was introduced in [400] where adversarial patches are designed to attack person detection. DAG algorithm [395] is an example of generating adversarial attacks for semantic segmentation and object detection tasks. It was discovered that the perturbations are exchangeable across different networks, even though they were trained differently since they share some intrinsic structure that makes them susceptible to a common source of perturbations. In addition to camera sensors, potential vulnerabilities of LiDAR-based autonomous driving detection systems are explored in [401]. Moreover, the 3D-printed adversarial objects showed effective physical attacks on LiDAR equipped vehicles, raising concerns about autonomous vehicles' safety. Robust Physical Perturbations (RP_2) [402] is another example that generates robust visual adversarial perturbations under different physical conditions on road sign classifications.

Defenses Against Adversarial Attacks

On the other hand, adversarial robustness and defense methods of neural networks have been studied to improve these networks' resistance to different adversarial attacks. One method for defense is adversarial training, where adversarial examples besides the clean examples are used to train the model. Adversarial training can be seen as a sort of simple data augmentation. Despite being simple, it cannot cover all attack cases. In [403], it is demonstrated that JPEG compression can undo the small adversarial perturbations created by the FGSM. However, this method is not adequate for large perturbations. Xu *et al*. [404] proposed Feature-squeezing for detecting adversarial examples. The model is tested on both the original input and the input after being pre-processed by feature squeezers such as spatial smoothing. If the output difference exceeds a certain threshold, we identify the input as an adversarial example. Defense-GAN [405] is another defense technique that employs generative adversarial networks (GAN)s [165], in which it seeks a similar output to a given picture while ignoring adversarial perturbations. It is shown to be a feasible defense that relies on the GAN's expressiveness and generative power. However, training GANs is still a challenging task. Robust attacks and defenses are still challenging tasks and an active area of research. Most previous works on adversarial attacks focused on single task scenarios. However, in real-life situations, multi-task learning is adopted to solve several tasks at once. Accordingly, multi-task networks leverage the shared knowledge among tasks, leading to better performance, reduced storage, and faster inference. Moreover, it is shown that when models are trained on multiple tasks at once, they become more robust to adversarial attacks on individual tasks [352]. However, defense remains an open challenge.

Chapter 4

Geometric Tasks

Contents

4.1 Problem Definition

In this chapter, the focus is on solving one of the most challenging perception problems for an autonomous car: *to predict distance of vehicles around it*. With the knowledge from **Chapter** 2 wherein we discussed the geometry of the camera models in detail, we incorporate the camera model into the core of the CNN framework. We present a novel self-supervised scale-aware framework for learning Euclidean distance and ego-motion by exploiting geometrical constraints in a sequence of images extracted from raw monocular fisheye videos without applying rectification. This work was formally presented as *FisheyeDistanceNet* [2] as an oral at the ICRA conference in 2020.

Building upon the success of this paper, we generalize the training framework to work with any camera model and propose a fully differentiable architecture that estimates the distance directly from raw unrectified images (shown in Figure 4.7) without the need for any pre-processing. This work was formally presented as *UnRectDepthNet* [3] as an oral at the IROS conference in 2020.

The method at the time of publication of these papers was state-of-the-art on KITTI and WoodScape datasets (refer Section 2.5) and showcased that it is possible to obtain distance maps on raw fisheye images without the need for rectification. This opens the door to rethink the need for rectification for fisheye images, and an in-detailed discussion about the motivation and problems encountered due to rectifying images is presented in Section 4.2.

4.2 Why is Predicting Depth so Difficult?

Before we dive deep into the depth estimation framework, we try to grasp some of the primary depth estimation issues in this section. Some of the complex challenges that must be solved include correspondence matching, which can be difficult due to factors such as textureless regions, occlusion, non-Lambertian surfaces, and resolving ambiguous solutions, in which several 3D scenes can give the same scene on the image plane, implying that estimated depth is not unique. The key culprit is the loss of depth information when 3D views are projected to 2D images. Another issue arises when there are motion and dynamic objects. *Lambertian surfaces* are those that tend to have the same brightness regardless of where they are viewed from. Owing to non-ideal diffuse reflection, the resulting brightness intensity in pictures depicting the same scene from two different perspectives may not be equal. *Occlusion* occurs when an object is occluded in one view but not the other, and *textureless region* occurs when several pixels have the same pixel intensity. The ability to retrieve distance information from a camera is very appealing due to its low production cost and dense representation. For the time being, the best alternative way to retrieve depth is to use an active range sensor such as Light Detection and Ranging (LiDAR). They are naturally high precision sensors that provide highly accurate depth information [112].

Depth Estimation is an Ill-Posed Problem

Many authors [52, 53, 55] would note that the problem of estimating depth from a single RGB image is an ill-posed inverse problem when researching monocular depth estimation. *i.e.*, several 3D scenes observed in the world may also correspond to the same 2D plane as shown in Figures 4.1 and 4.2. Figure 4.1 illustrates that (a) A line drawing provides information only about the x, y coordinates of points lying along the object contours. (b) The human visual system is usually able to reconstruct an object in three dimensions given only a single 2D projection (c) Any planar line drawing is geometrically consistent with infinitely many 3D structures.

Scale-Ambiguity for Monocular Depth Estimation: For any given camera model, adjusting the focal length will proportionately scale the points on the image plane (see Figure 4.2). Let us suppose we scale the entire scene points X, by a factor k and, at the same time, scale the camera matrices P, by a factor of $1/k$, the projections of

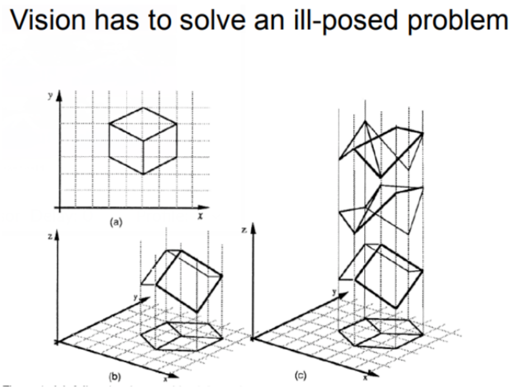


Figure 4.1 **An ill-posed problem in vision.** Figure reproduced from [407].

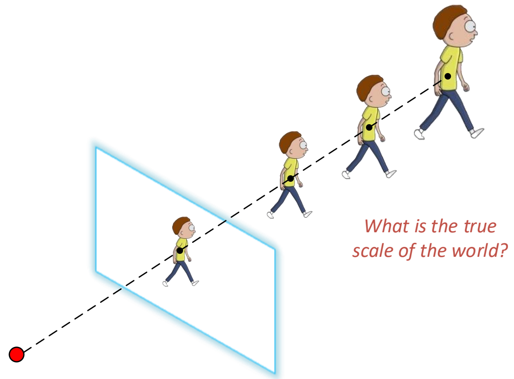


Figure 4.2 **Illustration of scale-ambiguity in depth estimation.**

the scene points in the image remain exactly the same *i.e.*,

$$
\begin{aligned}
x &= PX \\
&= \frac{1}{k} P * kX = x
\end{aligned}
\tag{4.1}
$$

Eq. 4.1 depicts that we can never recover the exact scale of the actual scene from the image alone with monocular approaches. This does not preclude us from making specific predictions, but it does suggest that all depth values would be relative to one another. As a result, an absolute value is needed to serve as an anchor point, a measurement from another dedicated sensor, in order to obtain actual depth estimation. The depth can then be calculated by dividing the approximate anchor value by the measured one. Some of the possible solutions are:

- Using an external sensor, such as LiDAR, Time-of-Flight, or stereo cameras, to measure the distance at least in one point. It is not a simple solution, and it necessitates embedded integration and accurate calibration to associate this calculation with the correct pixel in the image plane.
- If we assume depth consistency across training and testing dataset. It can be handy in datasets with high pose variability, such as KITTI [116], where the camera is *still* at the same height and looking at the ground from the same perspective. This assumption will fail for drones and unmanned aerial vehicles.
- We can measure the movement magnitude employing dedicated sensors, such as an IMU or GPS in a vehicle, or speed from the wheels in a car, and compare it to apparent calculated movement, assuming that both depth and apparent movement are consistently estimated. The scale factor would be the resulting ratio.

The last option is preferred because movement estimation is a critical calculation in navigation for autonomous vehicles, and it is thus almost always possible to obtain the speed information from the odometry data from a car [406].

Ill-pose: Projection ambiguity If we perform a geometric transformation on the scene as shown in Figure 4.3, these points would likely map to the same location on the plane after the transformation. Once again, we are faced with the same problem [112].

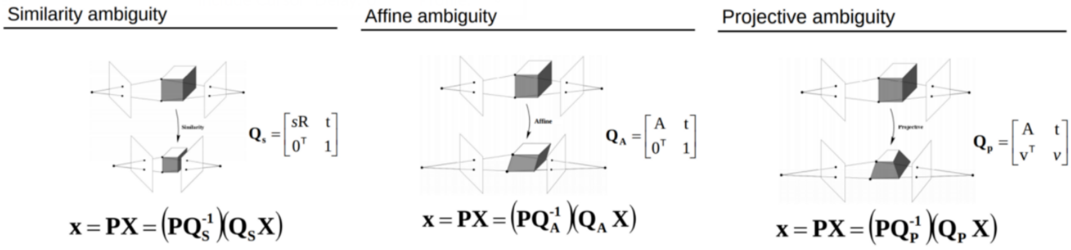

Similarity ambiguity
Similarity
$Q_s = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$
$x = PX = (PQ_S^{-1})(Q_S X)$
Affine ambiguity
Affine
$Q_A = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$
$x = PX = (PQ_A^{-1})(Q_A X)$
Projective ambiguity
Projective
$Q_p = \begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$
$x = PX = (PQ_P^{-1})(Q_P X)$

Figure 4.3 Projection of object after transformation maps to the same point in the plane. Figure reproduced from [409].

Dynamic Objects Violate Static World Assumption: Dynamic objects in the scene complicate the estimation process even more for the *SfM* framework. A moving camera and a series of static scenes are used to estimate depth via structure from motion. This assumption must hold true for pixel matching and alignment when there are moving objects in the scene, this assumption fails [112].

Why is Predicting Distance on Fisheye Cameras Even More Difficult?

Most state-of-the-art depth estimation works [52, 53, 54, 55] have solely focused on the moderate FoV/pinhole camera models as described in Section 2.1.2 wherein the networks are limited to work only on rectified image sequences. As discussed in Section 2.1 fisheye cameras undergo large distortions. The *SfM* framework and its core view synthesis approach are believed to work only on undistorted image pairs. For a pinhole projection model, $depth \propto 1/disparity$. Henceforth, the network's sigmoid output σ can be converted to depth with $D = 1/(a\sigma + b)$, where a and b are chosen to constrain D between 0.1 and 100 units [55]. For a spherical image, we can only obtain angular disparities [408] by rectification. To perform distance estimation on raw fisheye images, we would require metric distance values to warp the source image $I_{t'}$ onto the target frame I_t. Due to the limitations of the monocular *SfM* objective, both the monocular distance predictor g_d and ego-motion predictor g_x predict *scale-ambiguous* values as discussed in Section 4.2. This would make it impossible to estimate distance maps on fisheye images.

Distance estimation, especially in the context of autonomous vehicles, has proven to be difficult due to a variety of factors discussed earlier, such as occlusion, dynamic objects in the scene, and imperfect stereo correspondence. The biggest enemy for stereo matching algorithms is a reflective, translucent, or mirror surface. For example, the windshield of a car often degrades matching and thus results in erroneous estimation. As a result, most companies continue to rely on LiDAR to extract distance reliably. However, the latest trend in the autonomous vehicle perception stack goes towards sensor fusion since each sensor has a distinct advantage in its feature extraction methods. Nonetheless, since the emergence of Deep Learning, this field has gained significant momentum and achieved impressive results. Many studies have been conducted to address these problems [112].

In this thesis, we will solve some of the critical problems of the vision community by proposing novel solutions to tackle the monocular Distance estimation's scale factor issues and overcome the community's notion of employing *SfM* framework only on rectified sequences. We showcase that the *SfM* approach can be extended to raw distorted fisheye camera images.

Motivation for Working on Raw Fisheye Images

Rectification is considered to be a fundamental step in dense depth estimation [200]. In stereo cameras, epipolar rectification is performed to enable matching only in one direction along the horizontal scanline. This approach can also be extended to monocular cameras using two consecutive frames giving rise to motion stereo. These rectification steps also require the removal of non-linear distortion. Although it is convenient to work with rectilinear projections, there are practical issues that arise due to rectification. Rectification has also been transferred to CNN-based approaches as an inductive bias to simplify the learning. Yadati *et al*. [410] demonstrate that CNN-based two-view depth estimation is challenging without rectification and attempt to solve it in a more specific setting. To the best of our knowledge, all the methods reported on KITTI make use of barrel distortion corrected images. Automotive cameras such as fisheye surround-view cameras exhibit a strong distortion, and it is not easy to rectify their images. Recently, several tasks such as motion segmentation [21] and soiling detection [43] were demonstrated on fisheye images without rectification.

Practical Problems Encountered: Real-world automotive cameras have lens distortion, and the typical approach is to remove the distortion and then apply standard camera projection models. However, in practice, this has several issues that are not dealt with in literature. Figure 4.4 illustrates the rectification used in KITTI and WoodScape datasets. The first row shows a raw KITTI image with barrel distortion and the corresponding rectified image. The red box is used to crop out black pixels in the periphery, causing a loss in FoV. The second row shows a raw WoodScape image with strong fisheye lens distortion and the corresponding rectified image exhibiting a drastic loss of FoV. In the KITTI dataset, due to the barrel[1] distortion effect of the camera, images have been rectified and cropped to 1242 × 375 pixels. Cropping is performed after rectification to get a rectangular grid without any black pixels in the periphery. Thus, the rectified images' size is smaller than that of the raw images with 1392 × 512 pixels. Based on the number of the non-black, occupied pixels removed by the cropping, roughly 10% of the image information is lost. This effect becomes more drastic for the WoodScape images with a much larger radial distortion where more than 30% of the image information is lost[2]. For a horizontal FoV greater than 180°, there are rays incident from behind the camera, making it theoretically impossible to establish a complete mapping to a rectilinear viewport. Thus the rectification defeats the purpose of using a wide-angle fisheye lens.

Reduced FoV is the most critical problem of undistortion, but there are further practical issues. The first one is resampling distortion, which is caused by interpolation errors during the warping step. This effect can be partially mitigated by a more advanced interpolation method [411]. However, it is extreme in the periphery of fisheye lenses because a small region is expanded to a larger one in the warped image. Besides, the warping step is needed at inference time, which consumes significant computing power and memory bandwidth.

The other issue is related to calibration. In an industrial setup, millions of cameras are deployed, and they have manufacturing variations. The camera parameters (mainly focal length) can also vary due to high ambient temperatures when driving in a hot

[1] KITTI [116] refers to it as pincushion distortion because of an error in the OpenCV documentation, which was fixed later.

[2] Other quasi-linear rectification methods like cylindrical rectification will preserve more information at cost of additional distortion.

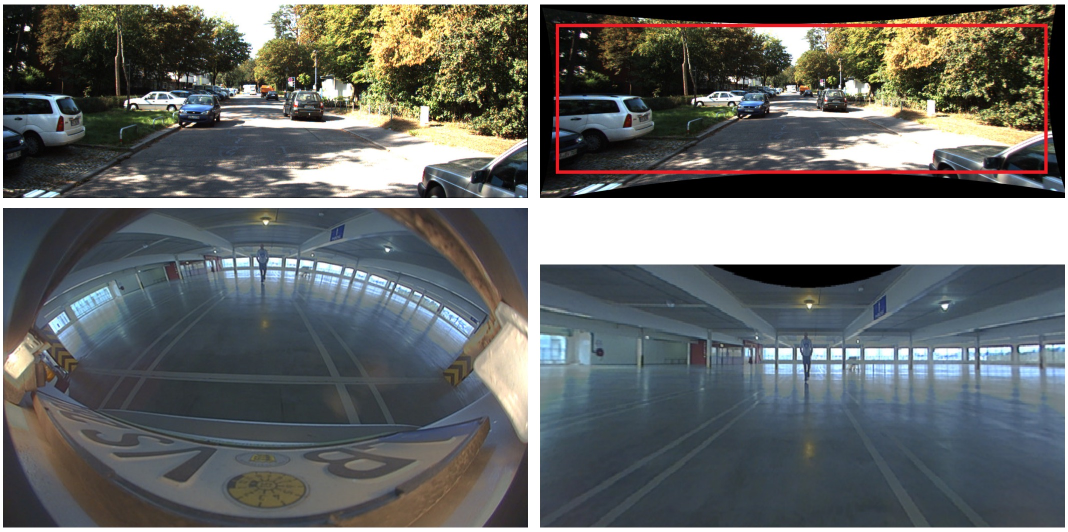


Figure 4.4 **Illustration of distortion correction in KITTI and WoodScape datasets.**

region. Thus a model that relies on rectification to correct the distortion could have errors. For instance, dataset capture and training are typically performed on one particular camera, and the model is deployed to work on millions of cameras in commercial vehicles. Thus rectification and cropping to a standard resolution as per the training camera are sub-optimal for a deployed camera. However, if CNN learns the distortion as part of the transfer function, it is only weakly encoded and expected to be more robust. To alleviate these issues, we are motivated to explore a distance estimation model that can work directly on raw images without needing rectification.

4.3 Self-Supervised Scale-Aware Distance Estimation Framework

Zhou *et al.*'s [53] self-supervised monocular *SfM* framework on which most self-supervised monocular depth estimation models build on, aims at learning:

1. a monocular distance model $g_d : I_t \to D$ predicting a scale-ambiguous distance $\hat{D} = g_d(I_t(p))$ per pixel p in the target image I_t; and
2. an ego-motion predictor $g_x : (I_t, I_{t'}) \to I_{t \to t'}$ predicting a set of six degrees of freedom of the rigid transformations $T_{t \to t'} \in \text{SE}(3)$, between the target image I_t and the set of reference images $I_{t'}$. Typically, $t' \in \{t+1, t-1\}$, *i.e.* the frames I_{t-1} and I_{t+1} are used as reference images, although using a larger window is possible.

This method is self-supervised because the ground truth is derived directly from the input signal—the RGB images in this case. There is no need for external data or signals to teach the network because the distance estimator g_d is its teacher! A limitation of this approach is that both distance and pose are estimated up to an unknown scale factor in the monocular *SfM* pipeline. One downside to this method is the scale ambiguity in both distance and pose estimation.

In this thesis, we recover scale-aware distance directly for distorted images (see Figure 4.5). The distance, which acts as an intermediary variable, is obtained from the network by constraining the model to perform image synthesis. As discussed earlier,

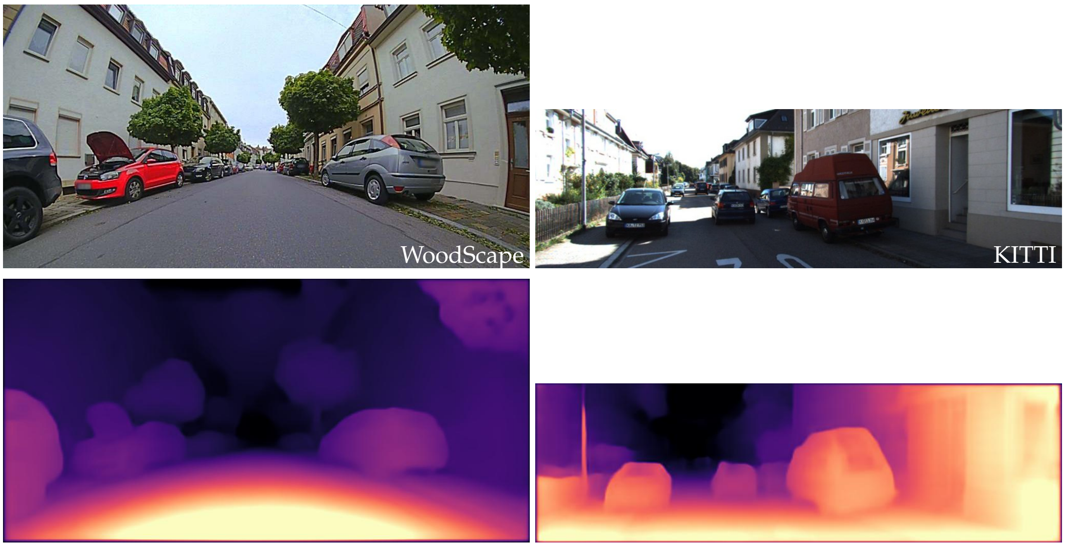


Figure 4.5 **Distance and depth derived from a single fisheye image (left) and single pinhole image (right) respectively.**

distance estimation is an ill-posed problem as there could exist a large number of possible incorrect distances per pixel, which can also recreate the novel view, given the relative pose between I_t and $I_{t'}$.

View-synthesis is used as a self-supervising technique, and the network is trained with the source images I_{t-1} and I_{t+1} to synthesize the appearance of a target image I_t on raw fisheye images. For this, we need the projection function Π of the chosen camera model, which maps a 3D point X_c in camera coordinates to a pixel $p = \Pi(X_c)$ in image coordinates. An overview of projection models for different lens types can be found in Section 2.1.1, and we specifically incorporate the Polynomial model. The corresponding unprojection function Π^{-1}, which maps an image pixel p and its distance estimate $\hat{D}$ to the 3D point $X_c = \Pi^{-1}(p, \hat{D})$, is also required. If Π^{-1} cannot be expressed in analytic form, a pre-calculated lookup table is used to ensure computational efficiency. A naive approach would be correcting raw fisheye images to piecewise or cylindrical projections and would essentially render the problem equivalent to Zhou *et al.*'s work [53]. In contrast, there is a simple yet efficient technique for obtaining scale-aware distance maps at the core of the approach. Figure 4.6 illustrates an overview of the *FisheyeDistanceNet* method.

This section discusses the geometry of the problem and how it is used to obtain differentiable losses. We describe the scale-aware *FisheyeDistanceNet* framework and its effects on the output distance estimates. Additionally, we provide an in-depth discussion of the various losses.

4.3.1 Modeling of Fisheye Geometry

Projection from Camera Coordinates to Image Coordinates

The projection function $X_c \mapsto \Pi(X_c) = p$ of a 3D point $X_c = (x_c, y_c, z_c)^T$ in camera coordinates to a pixel $p = (u, v)^T$ in the image coordinates is obtained via a 4th order

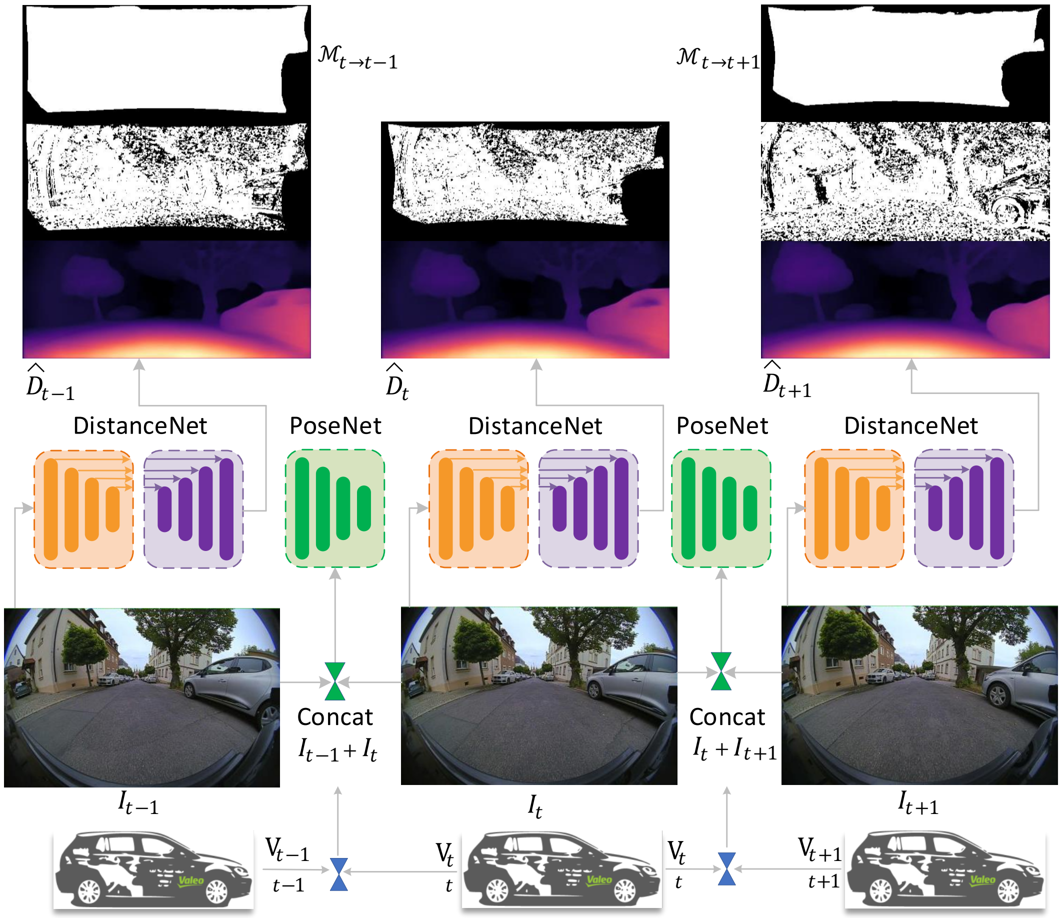


Figure 4.6 **Overview of the FisheyeDistanceNet framework.** The first row represents the ego masks as described in Section 4.3.4, $\mathcal{M}_{t\to t-1}$, $\mathcal{M}_{t\to t+1}$ indicate which pixel coordinates are valid when constructing $\hat{I}_{t-1\to t}$ from I_{t-1} and $\hat{I}_{t+1\to t}$ from I_{t+1} respectively. The second row indicates the masking of static pixels computed after two epochs, where black pixels are filtered from the photometric loss (*i.e.*, $\omega = 0$). It prevents dynamic objects at a similar speed as the ego car and low texture regions from contaminating the loss. The masks are computed for forward and backward sequences from the input sequence S and reconstructed images using Eq. 4.11 as described in Section 4.3.4. The third row represents the distance estimates corresponding to their input frames. The fourth row contains the encoder-decoder network architecture for distance and pose estimation. Finally, the vehicle's odometry data is used to resolve the scale factor issue.

polynomial in the following way:

$$\varphi = \operatorname{arctan2}(y_c, x_c) \tag{4.2}$$

$$\theta = \frac{\pi}{2} - \operatorname{arctan2}(z_c, r_c) \tag{4.3}$$

$$\varrho(\theta) = k_1 \cdot \theta + k_2 \cdot \theta^2 + k_3 \cdot \theta^3 + k_4 \cdot \theta^4 \tag{4.4}$$

$$p = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \varrho(\theta) \cdot \cos\varphi \cdot a_x + c_x \\ \varrho(\theta) \cdot \sin\varphi \cdot a_y + c_y \end{pmatrix} \tag{4.5}$$

where $r_c = \sqrt{x_c^2 + y_c^2}$, θ is the angle of incidence, $\varrho(\theta)$ is the mapping of incident angle to image radius, (a_x, a_y) is the aspect ratio and (c_x, c_y) is the principal point.

Unprojection from Image Coordinates to Camera Coordinates

The unprojection function $(p, \hat{D}) \mapsto \Pi^{-1}(p, \hat{D}) = X_c$ of an image pixel $p = (u, v)^T$ and it's distance estimate $\hat{D}$ to the 3D point $X_c = (x_c, y_c, z_c)^T$ is obtained via the following steps. Letting $(x_i, y_i)^T = \left((u - c_x)/a_x, (v - c_y)/a_y\right)^T$, we obtain the angle of incidence θ by numerically calculating the 4th order polynomial roots of $\varrho = \sqrt{x_i^2 + y_i^2}$ using the distortion coefficients k_1, k_2, k_3, k_4 (see Eq. 4.4). For training efficiency, we pre-calculate the roots and store them in a lookup table for all the pixel coordinates. Now, θ is used to get

$$r_c = \hat{D} \cdot \sin\theta \quad \text{and} \quad z_c = \hat{D} \cdot \cos\theta \tag{4.6}$$

where the distance estimate $\hat{D}$ from the network represents the Euclidean distance $\|X_c\| = \sqrt{x_c^2 + y_c^2 + z_c^2}$ of a 3D point X_c. The polar angle φ and the x_c, y_c components can be obtained as follows:

$$\varphi = \text{arctan2}(y_i, x_i), \;\; x_c = r_c \cdot \cos\varphi, \;\; y_c = r_c \cdot \sin\varphi.$$

4.3.2 Photometric Loss

Let us consider the image reconstruction error from a pair of images $I_{t'}$ and I_t, distance estimate $\hat{D}_t$ at time t, and the relative pose for I_t, with respect to the source image $I_{t'}$'s pose, as $T_{t \to t'}$. Using the distance estimate $\hat{D}_t$ of the network a point cloud P_t is obtained via:

$$P_t = \Pi^{-1}(p_t, \hat{D}_t) \tag{4.7}$$

where Π^{-1} represents the unprojection from image to camera coordinates as explained in Section 4.3.1, p_t the pixel set of image I_t. The pose estimate $T_{t \to t'}$ from the pose network is used to get an estimate $\hat{P}_{t'} = T_{t \to t'} P_t$ for the point cloud of the image $I_{t'}$. $\hat{P}_{t'}$ is then projected onto the fisheye camera at time t' using the projection model Π described in Section 4.3.1. Combining transformation and projection with Eq. 4.7 establishes a mapping from image coordinates $p_t = (u, v)^T$ at time t to image coordinates $\hat{p}_{t'} = (\hat{u}, \hat{v})^T$ at time t'. This mapping allows for the reconstruction $\hat{I}_{t' \to t}$ of the target frame I_t by backward warping the source frame $I_{t'}$.

$$\hat{p}_{t'} = \Pi\big(T_{t \to t'} \Pi^{-1}(p_t, \hat{D}_t)\big), \quad \hat{I}^{uv}_{t' \to t} = \left\langle I^{\hat{u}\hat{v}}_{t'} \right\rangle \tag{4.8}$$

Since the warped coordinates $\hat{u}, \hat{v}$ are continuous, we apply the differentiable spatial transformer network introduced by [412] to compute $\hat{I}_{t' \to t}$ by performing bilinear interpolation of the four pixels from $I_{t'}$ which lie close to $\hat{p}_{t'}$. The symbol $\langle \ldots \rangle$ denotes the corresponding sampling operator.

Following [54, 413] the image reconstruction error between the target image I_t and the reconstructed target image $\hat{I}_{t' \to t}$ is calculated using the L1 pixel-wise loss term combined with Structural Similarity (SSIM) [414], as the photometric loss $\mathcal{L}_p$ given by Eq. 4.9 below.

$$\begin{aligned} \tilde{\mathcal{L}}_p(I_t, \hat{I}_{t' \to t}) &= \alpha \, \frac{1 - \text{SSIM}(I_t, \hat{I}_{t' \to t}, \mathcal{M}_{t \to t'})}{2} \\ &\quad + (1 - \alpha) \, \left\| (I_t - \hat{I}_{t' \to t}) \odot \mathcal{M}_{t \to t'} \right\|_{l^1} \\ \mathcal{L}_p &= \min_{t' \in \{t+1, t-1\}} \tilde{\mathcal{L}}_p(I_t, \hat{I}_{t' \to t}) \end{aligned} \tag{4.9}$$

where $\alpha = 0.85$, $\mathcal{M}_{t\to t'}$ is the binary mask as discussed in Section 4.3.4 and the symbol $\odot$ denotes element-wise multiplication. Following [55] instead of averaging the photometric error over all source images, we adopt per-pixel minimum. This significantly sharpens the occlusion boundaries and reduces the artifacts resulting in higher accuracy.

The self-supervised framework assumes a static scene, no occlusion, and change of appearance (*e.g.*, brightness constancy). A large photometric cost is incurred, potentially worsening the performance if dynamic objects and occluded regions exist. These areas are treated as outliers similar to [415] and clip the photometric loss values to a 95^{th} percentile. Zero gradients are obtained for errors larger than 95%. This improves the optimization process and provides a way to strengthen the photometric error.

4.3.3 Solving the Scale Factor Ambiguity

To overcome the limitations of this *SfM* framework as discussed in Section 4.2 and to achieve scale-aware distance values, we normalize the pose network's estimate $T_{t\to t'}$ and scale it with Δx, the displacement magnitude relative to target frame I_t which is calculated using vehicle's instantaneous velocity estimates $v_{t'}$ at time t' and v_t at time t. We also apply this technique on KITTI [116] to obtain metric depth maps.

$$\overline{T}_{t\to t'} = \frac{T_{t\to t'}}{\|T_{t\to t'}\|} \cdot \Delta x \tag{4.10}$$

4.3.4 Masking Static Pixels and Ego Mask

Following [55], we incorporate a masking approach to filter out static pixels that do not change their appearance from one frame to the other in the training sequence. The approach would filter out objects that move at the same speed as the ego-car and ignore the static frame when the ego-car stops moving. Similar to other approaches [53, 55, 157, 363] the per-pixel mask ω is applied to the loss by weighting the pixels selectively. Instead of being learned from the object motion [189], the mask is computed in the forward pass of the network, yielding a binary mask output where $\omega \in \{0,1\}$. Wherever the photometric error of the warped image $\hat{I}_{t'\to t}$ is not lower than that of the original unwarped source frame $I_{t'}$ in each case compared to the target frame I_t, ω is set to ignore the loss of such pixels, *i.e.*

$$\omega = \big[\, \min_{t'} pe(I_t, \hat{I}_{t'\to t}) < \min_{t'} pe(I_t, I_{t'}) \,\big] \tag{4.11}$$

where $[\,]$ is the Iverson bracket. Additionally, we add a binary ego mask $\mathcal{M}_{t\to t'}$ proposed in [154] that ignores computing the photometric loss on the pixels that do not have a valid mapping *i.e.* some pixel coordinates of the target image I_t may not be projected onto the source image $I_{t'}$ given the estimated distance $\hat{D}_t$.

4.3.5 Edge-Aware Smoothness Loss

In order to regularize distance and avoid divergent values in occluded or textureless low-image gradient areas, we add a geometric smoothing loss. We adopt the edge-aware term similar to [54, 154, 416]. The regularization term is imposed on the inverse distance map. Unlike previous works, the loss is not decayed for each pyramid level by a factor of 2 due to down-sampling, as we use a super resolution network (see Section 4.4)

$$\mathcal{L}_s(\hat{D}_t) = |\partial_u \hat{D}_t^*| e^{-|\partial_u I_t|} + |\partial_v \hat{D}_t^*| e^{-|\partial_v I_t|} \tag{4.12}$$

To discourage shrinking of estimated distance [177], mean-normalized inverse distance of I_t is considered, i.e. $\hat{D}_t^* = \hat{D}_t^{-1}/\overline{D}_t$, where $\overline{D}_t$ denotes the mean of $\hat{D}_t^{-1} := 1/\hat{D}_t$.

4.3.6 Cross-Sequence Distance Consistency Loss

The *SfM* setting uses an N-frame training snippet $S = \{I_1, I_2, \cdots, I_N\}$ from a video as input. The *FisheyeDistanceNet* can estimate the distance of each image in the training sequence. Another constraint can be enforced among the frames in S since the distances of a 3D point estimated from different frames should be consistent.

Let us assume $\hat{D}_{t'}$ and $\hat{D}_t$ are the estimates of the images $I_{t'}$ and I_t respectively. For each pixel $p_t \in I_t$, we can use Eq. 4.8 to obtain $\hat{p}_{t'}$. Since it's coordinates are real valued, we apply the differentiable spatial transformer network introduced by [412] and estimate the distance value of $\hat{p}_{t'}$ by performing bilinear interpolation of the four pixel's values in $\hat{D}_{t'}$ which lie close to $\hat{p}_{t'}$. Let us denote the distance map obtained through this as $\hat{D}_{t\to t'}(p_t)$. Next, we can transform the point cloud in frame I_t to frame $I_{t'}$ by first obtaining P_t using Eq. 4.7. We transform the point cloud P_t using the pose network's estimate via $\hat{P}_{t'} = T_{t\to t'} P_t$. Now, $D_{t\to t'}(p_t) := \|\hat{P}_{t'}\|$ denotes the distance generated from point cloud $\hat{P}_{t'}$. Ideally, $D_{t\to t'}(p_t)$ and $\hat{D}_{t\to t'}(p_t)$ should be equal. Therefore, we can define the following cross-sequence distance consistency loss (CSDCL) for the training sequence S:

$$\begin{aligned}\mathcal{L}_{dc} = \sum_{t=1}^{N-1} \sum_{t'=t+1}^{N} \Bigg(& \sum_{p_t} \mathcal{M}_{t\to t'} \left| D_{t\to t'}(p_t) - \hat{D}_{t\to t'}(p_t) \right| \\ & + \sum_{p_{t'}} \mathcal{M}_{t'\to t} \left| D_{t'\to t}(p_{t'}) - \hat{D}_{t'\to t}(p_{t'}) \right| \Bigg)\end{aligned} \tag{4.13}$$

Eq. 4.13 contains one term for which pixels and point clouds are warped forwards in time (from t to t') and one term for which they are warped backwards in time (from t' to t).

In prior works [157, 416], the consistency error is limited to only two frames, whereas we apply it to the entire training sequence S. This induces more constraints and enlarges the baseline, inherently improving the distance estimation [415].

Backward Sequence

In the forward sequence, we synthesize the target frame I_t with the source frames I_{t-1} and I_{t+1} (*i.e.* as per above discussion $t' \in \{t+1, t-1\}$). Analogously, a backward sequence is carried out using I_{t-1} and I_{t+1} as target frames and I_t as source frame. We include warps $\hat{I}_{t\to t-1}$ and $\hat{I}_{t\to t+1}$, thereby inducing more constraints to avoid overfitting and resolve unknown distances in the border areas at the test time, as also observed in previous works [53, 55, 370]. We construct the loss for the additional backward sequence similar to the forward sequence. This comes at the cost of high computational effort and longer training time as we perform two forward and backward warps, which yields superior results on the WoodScape and KITTI dataset compared to the previous approaches [53, 55] which train only with one forward sequence and one backward sequence.

Depicting the importance of additional warps

The reconstructed image $\hat{I}^{uv}_{t'\to t}$ will result in a zoom-in operation where the border distance values are useful and will get meaningful gradients to train with. However, center values will have much noise due to the low displacement. On the other hand, since $\hat{I}^{uv}_{t\to t'}$ will result in a zoom-out effect, the border distance values are insignificant and should be filtered out from the photometric loss because the pixels that have to be retrieved do not exist. Center distance, though, will have a warp that sample values from the border of the frame, *i.e.*, with large displacement and the gradient will be less noisy than with $\hat{I}^{uv}_{t'\to t}$. Additional warps will induce more constraints to avoid overfitting and resolve unknown distances at the borders at test time.

4.3.7 Final Training Loss

The overall self-supervised *SfM* objective consists of a photometric loss $\mathcal{L}_p$ imposed between the reconstructed target image $\hat{I}_{t'\to t}$ and the target image I_t, included once for the forward and once for the backward sequence, and a distance regularization term $\mathcal{L}_s$ ensuring edge-aware smoothing in the distance estimates. Finally, $\mathcal{L}_{dc}$ a cross-sequence distance consistency loss derived from the chain of frames in the training sequence S is also included. To prevent the training objective from getting stuck in the local minima due to the gradient locality of the bilinear sampler [412], we adopt four scales to train the network as followed in [53, 54]. The final objective function is averaged per pixel, scale, and image batch.

$$\mathcal{L} = \sum_{n=1}^{4} \frac{\mathcal{L}_n}{2^{n-1}}, \tag{4.14}$$

$$\mathcal{L}_n = {}^{n}\mathcal{L}^{f}_{p} + {}^{n}\mathcal{L}^{b}_{p} + \gamma\, {}^{n}\mathcal{L}_{dc} + \beta\, {}^{n}\mathcal{L}_{s}$$

4.3.8 Handling Common Camera Distortion Models

Up to now, for the *SfM* framework to estimating depth, we employed a pinhole camera model on KITTI and a polynomial model WoodScape. Following the training regime's success, we illustrate an in-detail overview of the *UnRectDepthNet* [3]: A self-supervised generic training framework for handling common camera distortion models discussed in Section 2.1.1, to estimate depth directly from raw unrectified images is shown in Figure 4.7. The UnRectDepthNet training block on the right enables the usage of various camera models generically listed in the black box. The distortion is then handled internally in the unprojection and projection steps of the transformation from I_t to I_{t-1}. We test the generic framework with barrel distorted KITTI images and distorted WoodScape fisheye video sequences in this regime. The block on the left indicates the entire workflow of the training pipeline where the top row depicts the ego masks as explained in Section 4.3.4, $\mathcal{M}_{t\to t-1}$, $\mathcal{M}_{t\to t+1}$ represents the valid pixel coordinates while synthesizing $\hat{I}_{t-1\to t}$ from I_{t-1} and $\hat{I}_{t+1\to t}$ from I_{t+1} respectively. The following row showcases the masks used to filter static pixels obtained after training two epochs, and the black pixels are removed from the reconstruction loss. Dynamic objects moving at speeds similar to the ego car's and homogeneous areas are filtered out to prevent the erroneous signals in the reconstruction loss. The third row shows the depth predictions, where

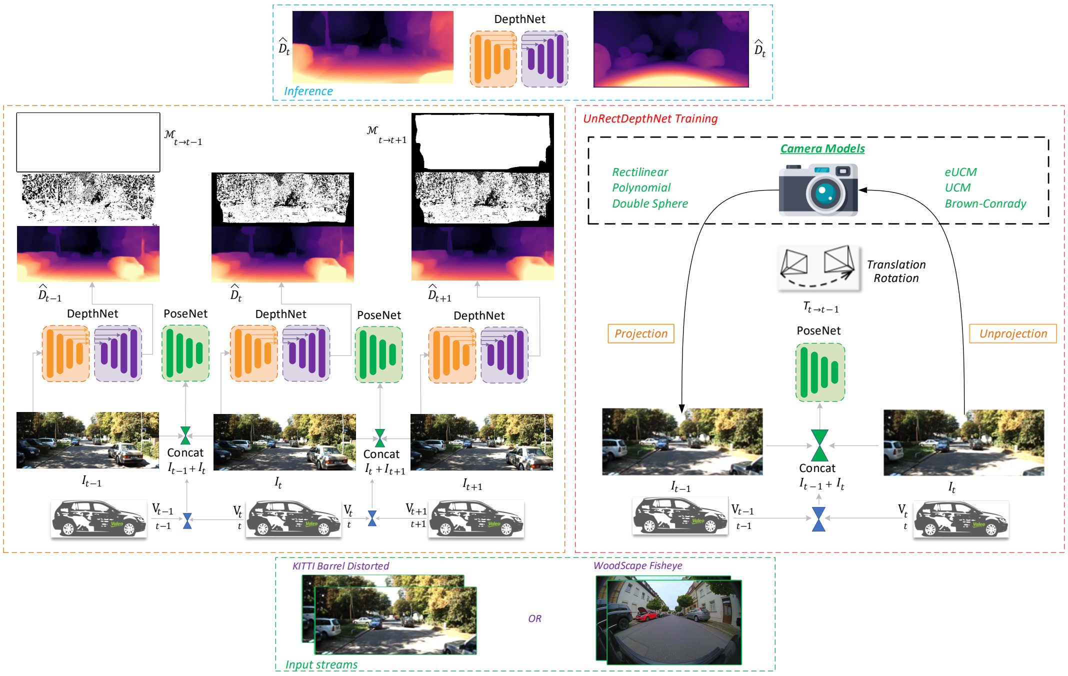


Figure 4.7 **Overview of the UnRectDepthNet: A generic depth estimation training framework that can handle various camera models.**

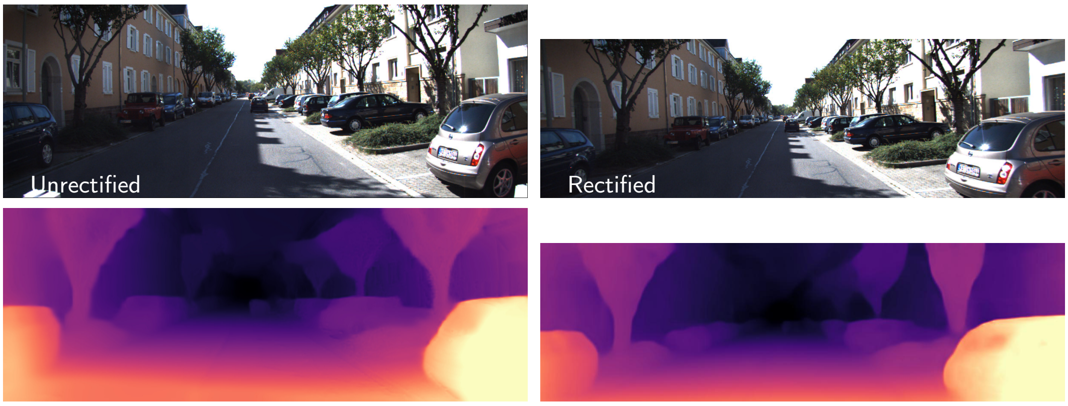


Figure 4.8 **Depth obtained from a single unrectified (left) and rectified KITTI image (right).**

the scale ambiguity is resolved using the ego vehicle's odometry data as discussed in Section 4.3.3. Finally, the top block illustrates the inference output. We compare a sample output from the framework on the unrectified (left) KITTI image vs. the standard rectified (right) KITTI image in Figure 4.8. We can see that our model handles the barrel distortion and outputs sharp depth maps without losing any FoV.

4.4 Network Details

The distance estimation network is mainly based on the U-net architecture [249], an *encoder-decoder* network with skip connections. After testing different ResNet family variants, such as ResNet50 with 25M parameters, we chose a ResNet18 [163] as the encoder. The key aspect here is replacing normal convolutions with deformable

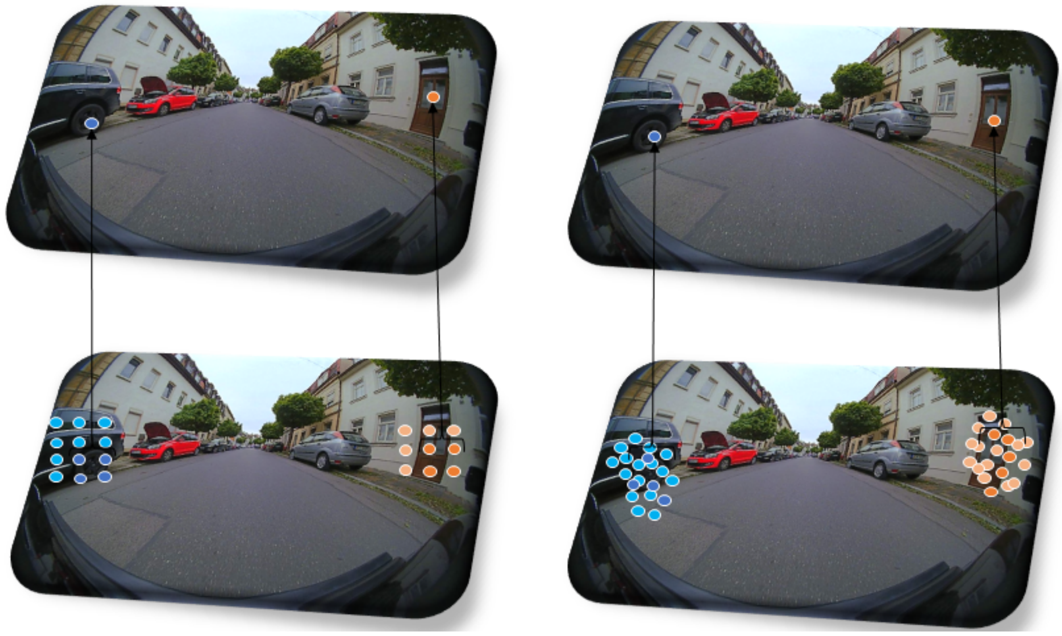


Figure 4.9 **Standard vs. Deformable convolution layer.** Deformable convolution will pick the values at different locations for convolutions conditioned on the input image.

convolutions since regular CNNs are inherently limited in modeling large, unknown geometric distortions due to their fixed structures, such as fixed filter kernels, fixed receptive field sizes, and fixed pooling kernels [207, 417].

In a deep CNN, the upper layers encode high-level scene information with weak spatial information, including object- or category-level evidence. Features from the middle layers are expected to describe middle-level representations of object parts and retain spatial information. Features from the lower convolution layers encode low-level spatial visual information like edges, corners, circles. That means that the middle and lower layers are responsible for learning spatial structures. If the deformable convolution is applied to the lower or middle layers, the spatial structures are susceptible to fluctuation. The spatial correspondence between input images and output distance maps is difficult to be preserved. This is the spatial correspondence problem indicated in [418], which is critical in pixel-wise distance estimation. Hence, deformable convolution is applied to the last few convolution layers as proposed by [207].

To alleviate this problem, Zhu *et al*. [417] proposed a better, more Deformable ConvNet (see Figure 4.9) with enhanced modeling power that can effectively model geometric transformations. We incorporate the enhanced modulated deformable convolutions to the *FisheyeDistanceNet* and *PoseNet*.

In previous works [53, 54, 55, 177, 370], the decoded features were upsampled via a nearest-neighbor interpolation or with learnable transposed convolutions. This process's main drawback is that it may lead to large errors at object boundaries in the upsampled distance map as the interpolation combines distance values of background and foreground. For effective and detailed preservation of the decoded features, we leverage the concept of sub-pixel convolutions [419] to the super-resolution network. We use pixel shuffle convolutions and replace the convolutional feature upsampling, performed via a nearest-neighbor interpolation or with learnable transposed convolutions. The resulting distance maps are super-resolved, have sharp boundaries, and expose more details of the scene.

The pose estimation network's backbone is based on [55] and predicts rotation using Euler angle parameterization. Compared to previous works [53, 154, 177], which stack the whole sequence as input and estimate the poses relative to the center image, we consider two consecutive images as input to the network, where I_t is the target view, and I_{t-1}, I_{t+1} are the source views. The output is a set of six DOF transformations between I_{t-1} and I_t as well as I_t and I_{t+1}. We replace standard convolutions with deformable convolutions for the encoder-decoder setting.

4.5 Experiments

4.5.1 Implementation Details

We use Pytorch [22] and employ the Ranger (RAdam [420] + LookAhead [421]) optimizer to minimize the training objective function (4.14). The model is trained using Titan RTX with a batch size of 20 for 20 epochs, with an initial learning rate of 4×10^{-4} with OneCycleScheduler [422]. The network's sigmoid output σ is converted to distance with $D = 1/(m \cdot \sigma + n)$ for pinhole model and $D = m \cdot \sigma + n$ for fisheye, where m and n are chosen such that D is bounded between 0.1 and 100 units. The fisheye image's original input resolution is 1280×800 pixels; we crop it to 1024×512 to remove the vehicle's bumper, shadow, and other artifacts of the vehicle. Finally, the cropped image is downscaled to 512×256 before feeding it to the network. For the pinhole model on KITTI, we use 640×192 pixels as the network input. We use 608×224 pixels as the network input to maintain the original aspect ratio for KITTI distorted images. The loss weighting factors β and γ of smoothness and cross-sequence distance consistency loss are set to 0.001. To remove checkerboard artifacts in the sub-pixel convolution [419], the final convolutional layers are initialized before the pixel shuffle operation as described in [423].

We experimented with batch normalization [424] and group normalization [425] layers in the encoder-decoder setting. We have found that group normalization with $G = 32$ significantly improves the results [426]. The smoothness weight term β and cross-sequence distance consistency weight term γ have been set to 0.001. We applied deformable convolutions to the 3 x 3 Conv layers in stages conv3, conv4, and conv5 in ResNet18 and ResNet50, with 12 layers of deformable convolution in the encoder part compared to 3 layers in [207], all in the conv5 stage for ResNet50. We replaced the subsequent layers of the decoder with deformable convolutions for the distance and pose network. For the pinhole model, on the KITTI Eigen split in Section 2.5.2, we used regular convolutions instead of deformable convolutions.

We evaluate *FisheyeDistanceNet* and *UnRectDepthNet* distance and depth estimation results using the metrics illustrated in Table 2.2 on KITTI and WoodScape datasets as described in Section 2.5 and report the results for less than 80 m as indicated in [50] for the pinhole model to facilitate a comparison. The quantitative results are shown in Tables 4.1 and 4.2 illustrates that the scale-aware self-supervised approach outperforms almost all the state-of-the-art monocular approaches. All the methods listed in the table are self-supervised approaches on monocular camera sequences. At inference time, all the approaches except *FisheyeDistanceNet*, *UnRectDepthNet*, and PackNet-SfM scale the estimated depths using median ground-truth LiDAR depth.

Approach	Abs Rel	Sq Rel	RMSE	RMSE$_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
	lower is better				higher is better		
KITTI							
Zhou [53]†	0.183	1.595	6.709	0.270	0.734	0.902	0.959
Yang [427]	0.182	1.481	6.501	0.267	0.725	0.906	0.963
Vid2depth [154]	0.163	1.240	6.220	0.250	0.762	0.916	0.968
GeoNet [370]†	0.149	1.060	5.567	0.226	0.796	0.935	0.975
DDVO [177]	0.151	1.257	5.583	0.228	0.810	0.936	0.974
DF-Net [416]	0.150	1.124	5.507	0.223	0.806	0.933	0.973
Ranjan [303]	0.148	1.149	5.464	0.226	0.815	0.935	0.973
EPC++ [363]	0.141	1.029	5.350	0.216	0.816	0.941	0.976
Struct2depth '(M)' [189]	0.141	1.026	5.291	0.215	0.816	0.945	0.979
Zhou [415]	0.139	1.057	5.213	0.214	0.831	0.940	0.975
PackNet-SfM [7]	0.120	0.892	4.898	0.196	0.864	0.954	0.980
Monodepth2 [55]	**0.115**	0.903	4.863	0.193	**0.877**	0.959	0.981
FisheyeDistanceNet	0.117	**0.867**	**4.739**	**0.190**	0.869	**0.960**	**0.982**
FisheyeDistanceNet (1024 × 320)	0.109	0.788	4.669	0.185	0.889	0.964	0.982
WoodScape							
FisheyeDistanceNet cap 80 m	0.167	1.108	3.814	0.216	0.794	0.953	0.972
FisheyeDistanceNet cap 40 m	0.152	0.768	2.723	0.210	0.812	0.954	0.974
FisheyeDistanceNet cap 30 m	0.149	0.613	2.402	0.204	0.810	0.957	0.976

Table 4.1 **Quantitative results of leaderboard algorithms on KITTI dataset [116] and FisheyeDistanceNet on WoodScape [12].** † marks newer results reported on GitHub.

	Method	Resolution	Dataset	Abs Rel	Sq Rel	RMSE	RMSE$_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
				lower is better				higher is better		
Original [50]	SfMLeaner [53]	416 x 128	K	0.183	1.595	6.709	0.270	0.734	0.902	0.959
	Vid2depth [154]	416 x 128	K	0.163	1.240	6.220	0.250	0.762	0.916	0.968
	DDVO [177]	416 x 128	K	0.151	1.257	5.583	0.228	0.810	0.936	0.974
	EPC++ [363]	640 x 192	K	0.141	1.029	5.350	0.216	0.816	0.941	0.976
	Struct2Depth [189]	416 x 128	K	0.141	1.026	5.291	0.215	0.816	0.945	0.979
	Monodepth2 [55]	640 x 192	K	0.115	0.903	4.863	0.193	0.877	0.959	0.981
	PackNet-SfM [7]	640 x 192	K	0.111	0.785	4.601	0.189	0.878	0.960	0.982
	Monodepth2 [55]	1024 x 320	K	0.115	0.882	4.701	0.190	0.879	0.961	0.982
	UnRectDepthNet	640 x 192	K	**0.107**	**0.721**	**4.564**	**0.178**	**0.894**	**0.971**	**0.986**
	UnRectDepthNet	1024 x 320	K	0.103	0.705	4.386	0.164	0.897	0.980	0.989
	UnRectDepthNet	608 x 224	KD	0.102	0.720	4.559	0.183	0.892	0.973	0.988
	UnRectDepthNet	1216 x 448	KD	0.106	0.709	4.357	0.161	0.895	0.984	0.992
	FisheyeDistanceNet [2]	512 x 256	WS	0.152	0.768	2.723	**0.210**	0.812	0.954	0.974
	UnRectDepthNet	512 x 256	WS	**0.148**	**0.702**	**2.530**	0.212	**0.826**	**0.960**	**0.980**
Improved [143]	SfMLeaner [53]	416 x 128	K	0.176	1.532	6.129	0.244	0.758	0.921	0.971
	Vid2Depth [154]	416 x 128	K	0.134	0.983	5.501	0.203	0.827	0.944	0.981
	DDVO [177]	416 x 128	K	0.126	0.866	4.932	0.185	0.851	0.958	0.986
	EPC++ [363]	640 x 192	K	0.120	0.789	4.755	0.177	0.856	0.961	0.987
	Monodepth2 [55]	640 x 192	K	0.090	0.545	3.942	0.137	0.914	0.983	0.995
	PackNet-SfM [7]	640 x 192	K	**0.078**	0.420	3.485	0.121	**0.931**	0.986	**0.996**
	UnRectDepthNet	640 x 192	K	0.081	**0.414**	**3.412**	**0.117**	0.926	**0.987**	**0.996**
	UnRectDepthNet	640 x 224	KD	0.092	0.458	3.503	0.132	0.906	0.971	0.990

Table 4.2 **Quantitative performance comparison of UnRectDepthNet** for depths up to 80 m for KITTI and 40 m for WoodScape. In the Dataset column, K refers to KITTI [11], KD refers to the KITTI distorted [116], and WS refers to WoodScape [12] dataset. *Original* refers to depth maps defined in [50], and *Improved* refers to refined depth maps provided by [143].

We generalized our previous model *FisheyeDistanceNet* in our new training framework and added additional features that improve results on WoodScape. For the fisheye dataset, we estimate distance rather than depth. The qualitative results are illustrated in Figures 4.10, 4.11 and 4.12 where the framework produces sharp distance maps on raw fisheye and pinhole images, respectively. The KITTI distorted results are better than most of the previous outcomes obtained with self-supervised approaches on the corresponding rectified dataset. We could not leverage the Cityscapes dataset into

Method	FS	BS	SR	CSDCL	DCN	Abs Rel	Sq Rel	RMSE	$RMSE_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Ours	✓	✓	✓	✓	✓	0.152	0.768	2.723	0.210	0.812	0.954	0.974
Ours	✓	✗	✓	✓	✓	0.172	0.829	2.925	0.243	0.802	0.952	0.970
Ours	✓	✗	✗	✓	✓	0.181	0.913	3.180	0.250	0.823	0.938	0.963
Ours	✓	✗	✗	✗	✓	0.190	0.997	3.266	0.258	0.796	0.930	0.963
Ours	✓	✗	✗	✗	✗	0.201	1.282	3.589	0.276	0.590	0.898	0.949

Table 4.3 **Ablation study on different variants of the FisheyeDistanceNet using the WoodScape dataset.** BS, SR, CSDCL, and DCN represent a backward sequence, super-resolution network with PixelShuffle, or sub-pixel convolution initialized to convolution NN resize (ICNR) [423], cross-sequence distance consistency loss, and deformable convolutions respectively.

the training regime to benchmark the scale-aware framework due to the absence of odometry data.

Since the projection operators are different, previous *SfM* approaches will not be feasible on the Woodscape dataset without adapting the network and projection model. It is important to note that due to the fisheye's geometry, it would not be a fair comparison to evaluate the distance estimates up to 80 m. The fisheye automotive cameras also undergo high data compression, and the dataset contains images of inferior quality compared with KITTI. The fisheye cameras can perform well up to a range of 40 m. Therefore, we also report results on a 30 m, and a 40 m range (see Table 4.1).

We generalized the training methodology of this model to incorporate any arbitrary distortion model. We also tuned the network to the optimal hyperparameters using grid search and removed batch normalization in the decoder as we observed ghosting effects and holes in homogeneous areas. We calculated the minimum reconstruction error for the two warps of the backward sequence individually compared to a combined minimization for forward sequences since here the target frames are $I_{t'}$ ($t' \in \{t+1, t-1\}$).

4.5.2 Fisheye Ablation Study

We conduct an ablation study to evaluate the importance of different components. We cap the distances at 40 m and the input resolution is 512 × 256 pixels. We ablate the following components and report their impact on the distance evaluation metrics in Table 4.3:

- *Remove Backward Sequence*: The network is only trained for the forward sequence, which consists of two warps as explained in Section 4.3.6. This has a critical impact on the model's performance as the induced baseline during training decreases due to fewer warps. The only advantage of this is lesser training time.
- *Additionally remove Super-Resolution using sub-pixel convolution*: Removal of sub-pixel convolution has a significant impact on Woodscape compared to KITTI. This is mainly attributed to the fisheye model, as far-away objects are tiny and cannot be resolved accurately with naive nearest-neighbor interpolation or transposed convolution [428].
- *Additionally remove cross-sequence distance consistency loss*: Removing the CSDCL mainly diminishes the baseline.

Method	FS	BS	SR	CSDCL	Abs Rel	Sq Rel	RMSE	RMSE$_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Ours	✓	✓	✓	✓	0.102	0.720	4.559	0.183	0.892	0.973	0.988
Ours	✓	✗	✓	✓	0.131	0.856	4.933	0.198	0.853	0.954	0.968
Ours	✓	✗	✗	✓	0.141	0.971	5.183	0.206	0.831	0.941	0.953
Ours	✓	✗	✗	✗	0.144	1.011	5.204	0.225	0.822	0.945	0.949

Table 4.4 **Ablation study of UnrectdepthNet on the KITTI dataset.** Depths are capped at 80 m. FS, BS, SR, CSDCL indicate forward sequence, backward sequence, super-resolution network with PixelShuffle [423] layers and cross-sequence depth consistency loss, respectively. The input resolution is 608 × 224 pixels.

- *Additionally remove deformable convolutions*: If we remove all the major components, especially deformable convolution layers [417], the model will fail miserably as the distortion introduced by the fisheye model will not be learned correctly by normal convolutional layers.

4.5.3 KITTI Distorted Ablation Study

We perform an ablation study to understand the significance of different components used and tabulate in Table 4.4:

- *Remove Backward Sequence*: The network is trained only for a forward sequence consisting of two warps, as explained in [2]. The impact is significant in the border areas as fewer constraints are induced. The model inherently fails to resolve unknown depths in those areas at the test time, which was also observed in previous works [53, 55, 370]. The only advantage of this is lesser training time.
- *Additionally remove Super-Resolution using sub-pixel convolution*: It has a significant effect as distant objects are small in fisheye cameras and cannot be resolved correctly with simple nearest-neighbor interpolation or transposed convolution [428].
- *Additionally remove cross-sequence depth consistency loss*: The removal of the CSDCL diminishes the baseline, induces fewer constraints, and the model is therefore not robust enough to yield accurate depth estimates.

4.6 Conclusion

This chapter presented two novel strategies in the field of self-supervised distance estimation. Firstly, a novel self-supervised training strategy to obtain metric distance maps on unrectified fisheye images. Secondly, a generic self-supervised training method for depth estimation handling distorted images. We showed that it is possible to support various commonly used automotive camera models in the framework and indicate empirical results on KITTI and WoodScape datasets. We show that the *FisheyeDistanceNet* and *UnRectDepthNet* establish a new state-of-the-art in the self-supervised monocular distance and depth estimation through extensive experiments on WoodScape and KITTI datasets, respectively. For KITTI, we show that depth estimation on unrectified images can produce the same accuracy as on rectified images. We obtain promising results, demonstrating the potential of using a CNN-based approach deployable in commercial automotive systems, particularly for replacing current classical depth estimation approaches.

In the following chapter, we will dig deeper into the choice of the self-supervised distance estimation loss functions and improve the model's robustness by incorporating semantic information. We will investigate how to leverage more directly the 2$^{\text{nd}}$ class of perception algorithm: Semantic Segmentation to guide geometric representation learning and induce a synergy between both the tasks for autonomous driving. We will further extend this approach on surround-view cameras, and with novel techniques, we aim for a large scale deployment of the model.

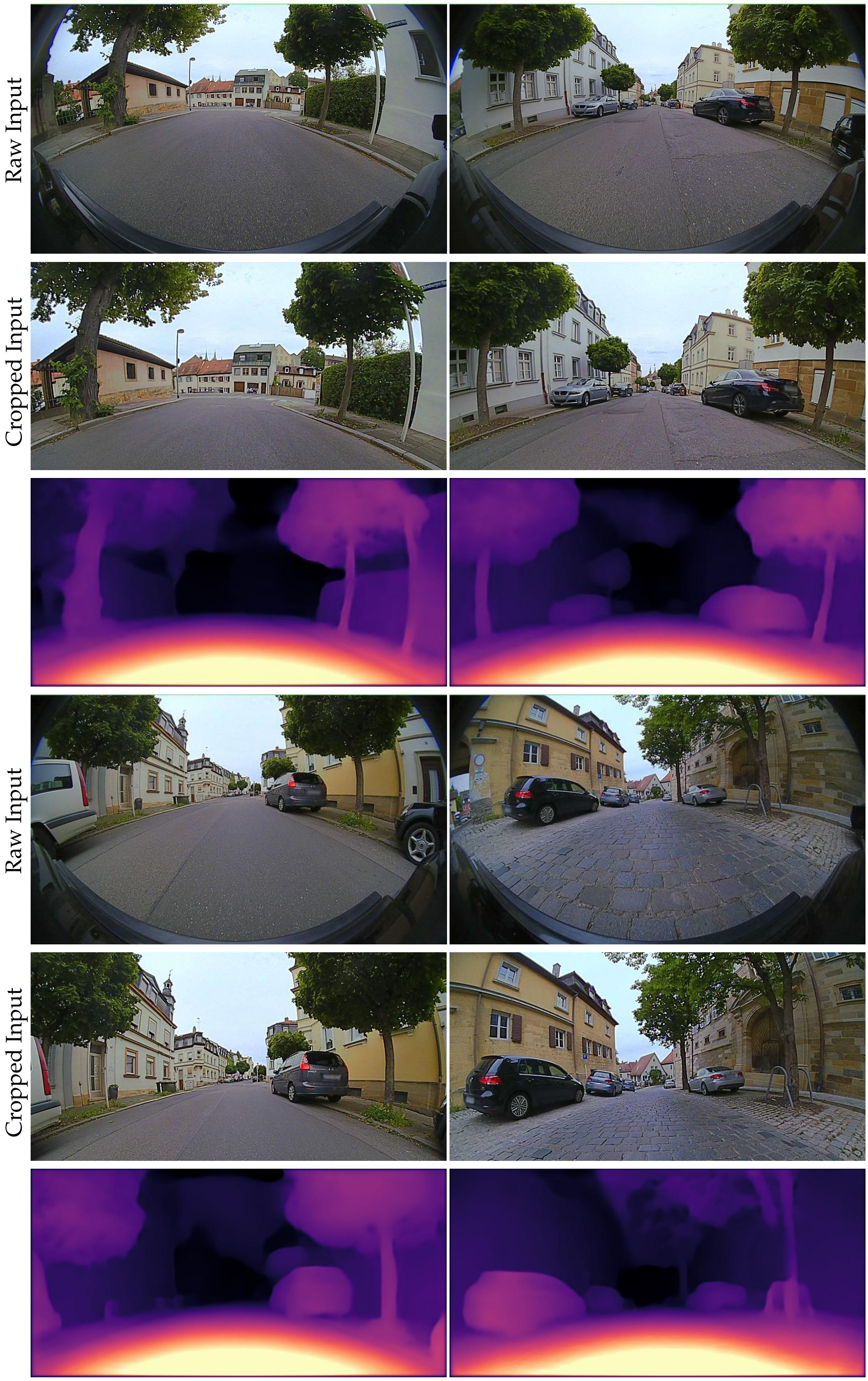


Figure 4.10 **Qualitative results of FisheyeDistanceNet on the WoodScape dataset.** For more qualitative results, we refer to this video: `https://youtu.be/Sgq1WzoOmXg`.

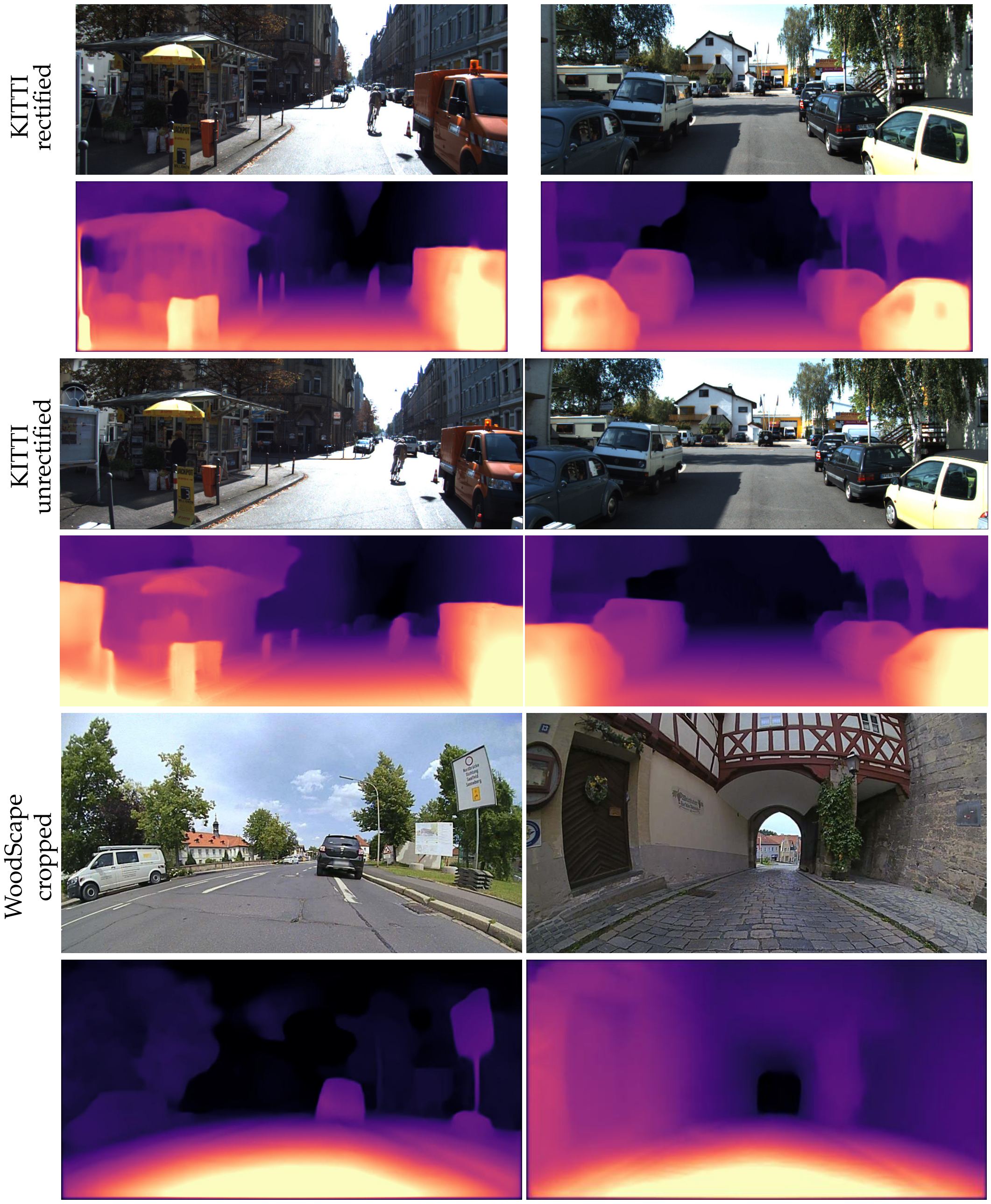


Figure 4.11 **Qualitative results comparison of UnRectDepthNet on KITTI and WoodScape dataset.** The results on a distorted test video sequence indicate excellent performance, see `https://youtu.be/K6pbx3bU4Ss`.

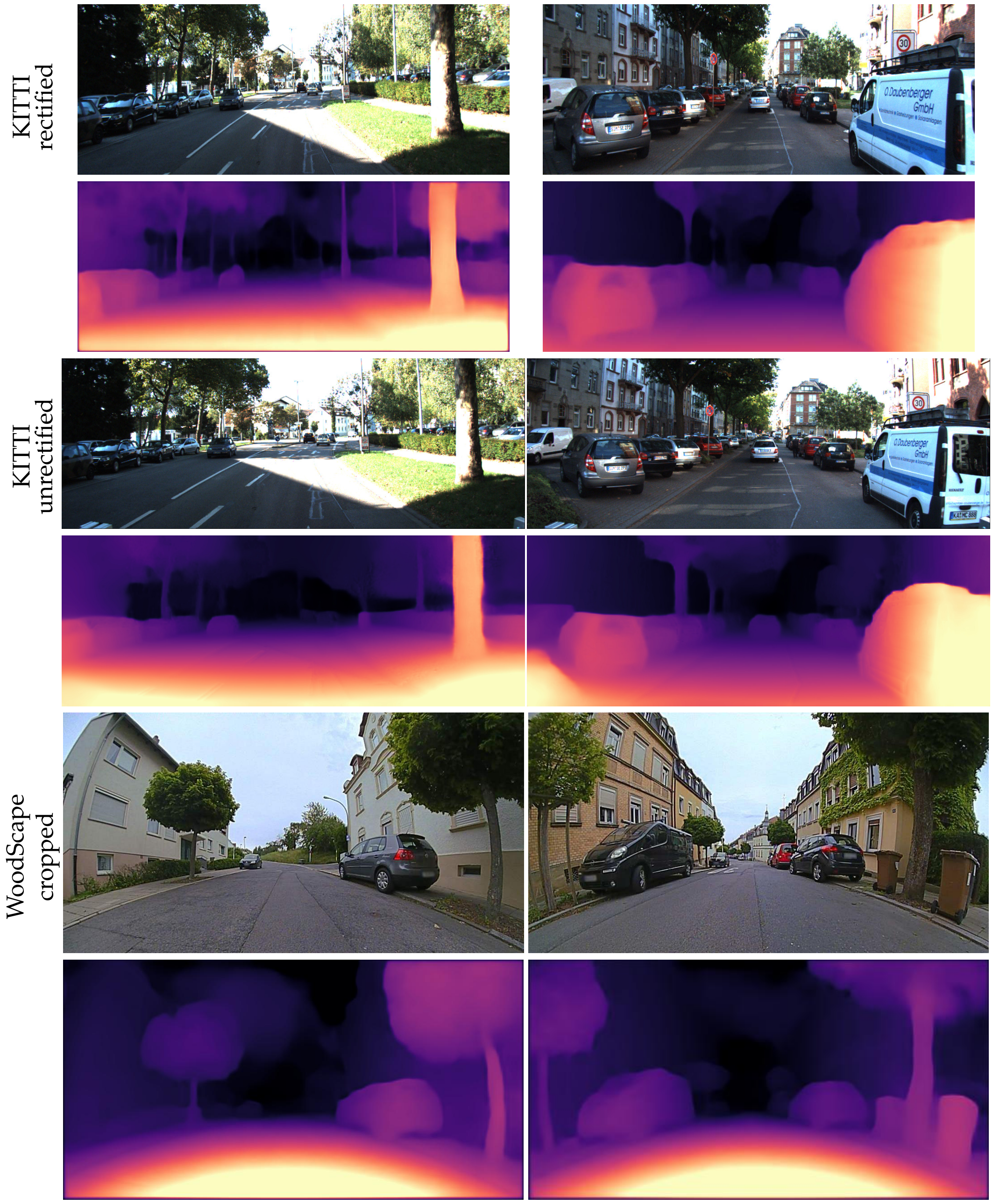


Figure 4.12 **Additional qualitative results comparison of UnRectDepthNet on KITTI and WoodScape dataset.**

Chapter 5

Geometry Meets Semantics

Contents

5.1 Problem Definition

As in **Chapter** 4, where we set up a *SfM* framework for fisheye images to predict distance maps by performing view-synthesis, in this chapter, we examine how to leverage more directly the *semantic* context of the scene to guide geometric representation learning while remaining in the self-supervised regime. Further, we dig deep into the loss functions and the architecture aspects for distance estimation and propose a near-field distance estimation solution on surround-view fisheye cameras targeting large-scale industrial deployment.

The first contribution in that sense is the novel application of a general and robust loss function proposed by [179] to the task of self-supervised distance estimation, which replaces the de facto standard of an L_1 loss function used in previous approaches [2, 3, 7, 55, 189]. The most important of all is a novel solution to filter out the dynamic objects from contaminating the photometric loss during training and the infinite distance issue during inference. This work was formally presented as *SyndistNet* [4] as an oral at the WACV conference in 2021.

Typically, automotive perception systems use multiple cameras, with current systems having at least four cameras. The number is likely to increase to more than ten cameras for future generation systems. Such surround-view cameras are focused on near field sensing, which is typically used for low-speed applications such as parking or traffic jam assistance functions [60]. The surround-view distance estimation framework will be facilitated by employing a single network on images from multiple cameras. A surround-view coverage of geometric information can be obtained for an autonomous vehicle by utilizing and post-processing the distance maps from all cameras.

Near-field distance estimation is a challenging problem because of distortion and partial visibility of close-by objects. Also, centimeter-level accuracy is required to enable precise low-speed maneuvers such as parking. Up to now, for the generation of high-quality distance estimates, one network per camera has to be trained, inducing unfeasible computational complexity with an increasing number of cameras. One of the thesis's main goals is to target the design of a model that can be deployed in millions of vehicles having its own set of cameras. To do so, we present a novel camera geometry adaptive multi-scale convolution to incorporate the camera parameters into the self-supervised distance estimation framework. This work was very influential from a product perspective to win next-generation projects and be influential in the academic community. This work formally presented as *SVDistNet* [5] to a journal at the T-ITS in 2021.

As depicted in Figure 5.1, dynamic objects induce a lot of unfavorable artifacts and hinder the photometric loss during the training, which results in infinite distance predictions, *e.g.*, due to their violation of the static world assumption. Therefore, we use the segmentation masks to apply a simple semantic masking technique, based on the temporal consistency of consecutive frames, which delivers significantly improved results, *e.g.*, concerning the infinite distance problem of objects moving at the same speed as the ego-camera. Compared to previous approaches, the semantically guided distance estimation in Figure 5.1 produces sharper depth edges and reasonable distance estimates for dynamic objects. Previous approaches [303, 363] did predict these motion masks only implicitly as part of the projection model and therefore were limited to the projection model's fidelity.

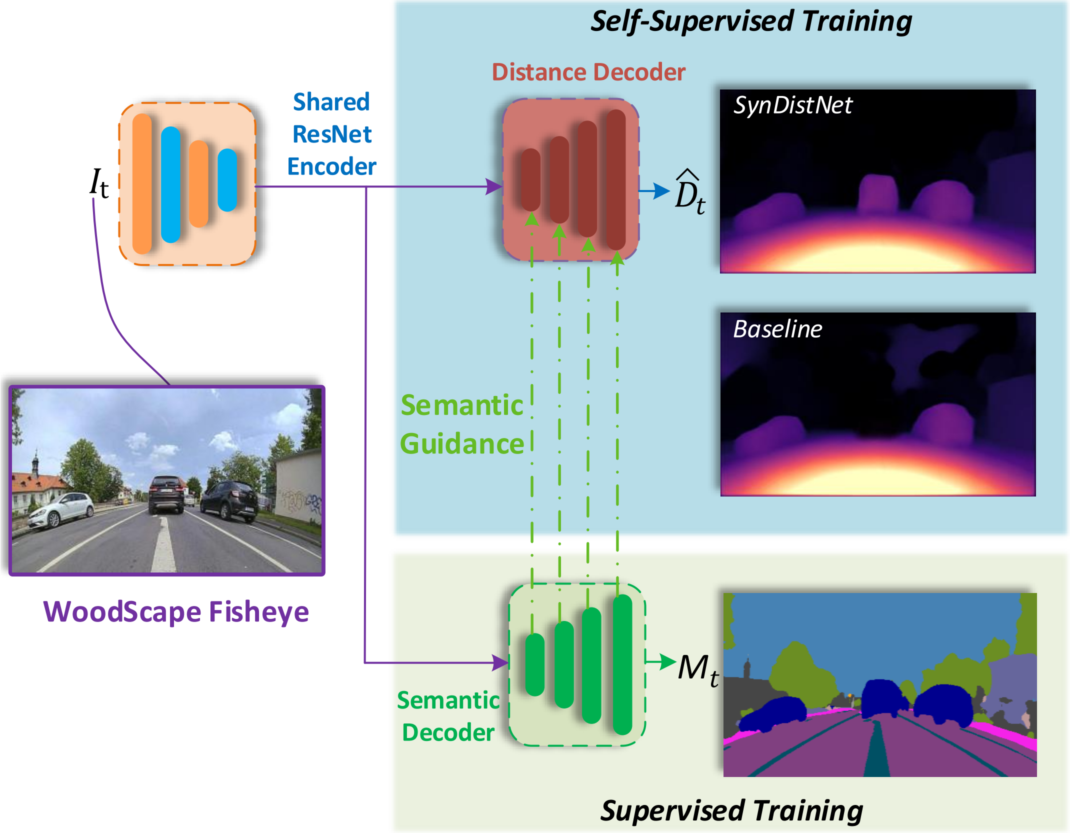


Figure 5.1 **Overview of the SynDistNet framework and a comparison with FisheyeDistanceNet baseline.** We jointly predict the distance $\hat{D}_t$ and the semantic segmentation M_t from a single input image I_t.

5.2 Multi-Task Learning Framework

This section describes the framework for the multi-task learning of distance estimation and semantic segmentation as illustrated in Figure 5.2. The upper part (blue blocks) describes the single steps required for distance estimation, while the green blocks describe the single steps needed to predict semantic segmentation. Both tasks are optimized inside a multi-task network by using the weighted total loss described in Eq. 5.8. Following Section 4.3, we set up the self-supervised monocular structure-from-motion (SfM) framework with extensions to enabling multiple cameras. View synthesis is performed by incorporating the polynomial projection model from Section 4.3.1. The same protocols are used to train the distance and pose estimation networks simultaneously. Section 4.3.7 describes the total self-supervised objective loss. In the following part, we will describe the different loss contributions in the context of fisheye camera images.

5.2.1 Semantic Segmentation Baseline

We define semantic segmentation as the task of assigning a pixel-wise label mask M_t to an input image I_t, i.e. the same input as for distance estimation from a single image. Each pixel gets assigned a class label $s \in \mathcal{S} = \{1, 2, ..., S\}$ from the set of classes $\mathcal{S}$. In a supervised way, the network predicts a posterior probability Y_t that a pixel belongs to a class $s \in \mathcal{S}$, which is then compared to the one-hot encoded ground truth labels

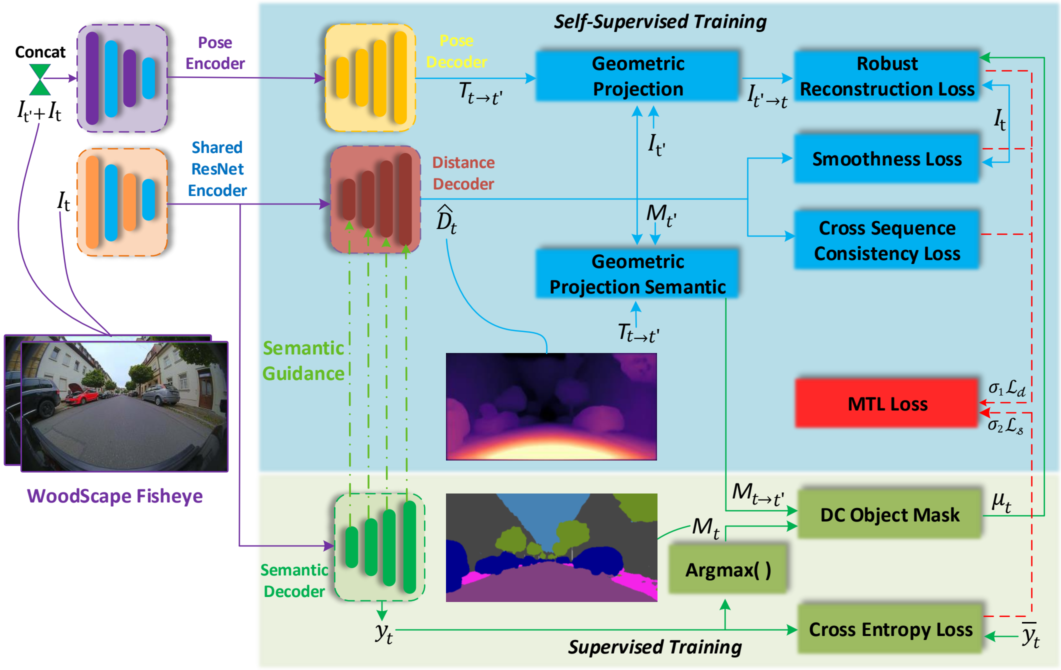


Figure 5.2 **Overview of the SynDistNet framework** for the joint prediction of distance and semantic segmentation.

$\overline{Y}_t$ inside the cross-entropy loss

$$\mathcal{L}_{ce} = -\sum_{s\in\mathcal{S}} \overline{Y}_{t,s} \cdot \log\left(Y_{t,s}\right) \tag{5.1}$$

the final segmentation mask M_t is then obtained by applying a pixel-wise argmax operation on the posterior probabilities $Y_{t,s}$. Note that we also use unrectified fisheye camera images, for which the segmentation task can however still be applied as shown in this work.

5.2.2 Robust Reconstruction Loss

Most state-of-the-art self-supervised depth estimation methods use heuristic loss functions. However, the optimal choice of a loss function is not well defined theoretically. In this section, we emphasize the need for the exploration of a better photometric loss function described in Section 4.3.2 and explore a more generic robust loss function.

Towards developing a more robust loss function, we introduce the common notion of a per-pixel regression ρ in the context of distance estimation, which is given by

$$\rho\left(\xi\right) = \rho\left(\hat{I}_{t'\to t} - I_t\right) \tag{5.2}$$

while this general loss function can be implemented by a simple L_1 loss as in the second term of Eq. 4.9, recently, a general and more robust loss function has been proposed by Barron [179], which we use to replace the L_1 term in Eq. 4.9. This function is a generalization of many common losses such as the L_1, L_2, Geman-McClure, Welsch/Leclerc, Cauchy/Lorentzian and Charbonnier loss functions. In this loss, robustness is introduced as a continuous parameter and it can be optimized within the loss function to improve the performance of regression tasks. The general form of the loss function is:

$$f_{\text{rob}}\left(\zeta,\rho,c\right)=\frac{|\rho-2|}{\rho}\left(\left(\frac{\left(\zeta/c\right)^2}{|\rho-2|}+1\right)^{\rho/2}-1\right) \tag{5.3}$$

The free parameters in this loss function can be automatically adapted to any particular problem via data-driven optimization. To induce ρ as a trainable parameter Barron [179] encapsulates the loss into a probability density function given by:

$$p\left(\zeta\mid\mu,\rho,c\right)=\frac{1}{cZ\left(\rho\right)}\exp\left(-\rho\left(\zeta-\mu,\rho,c\right)\right) \tag{5.4}$$

$$Z\left(\rho\right)=\int_{-\infty}^{\infty}\exp\left(-\rho\left(\zeta,\rho,1\right)\right) \tag{5.5}$$

where $p\left(\zeta\mid\mu,\rho,c\right)$ is only defined if $\rho\geq 0$, as $Z\left(\rho\right)$ is divergent when $\rho<0$. Then the optimization function reduces to:

$$\arg\min_{\theta,\rho}-log(p(\zeta|\rho)=\rho\left(\zeta,\rho\right)+log(Z(\rho)) \tag{5.6}$$

where $log(Z(\rho))$ is an analytical function which is approximated with a cubic spline function. $Z\left(\rho\right)$ is an important factor in the loss function as it reduces the cost of outliers. The loss of outliers decreases with the reduction of ρ. Correspondingly, the loss of inliers will increase. The main properties of the robust loss function are (i) It is monotonic with respect to its inputs $|\zeta|$ and ρ which is useful for graduated non-convexity. (ii) It is smooth respect to its inputs ζ and ρ (*i.e.*, in C^{∞}). (iii) It has bounded first and second derivatives (no exploding gradients and easier pre-conditioning).

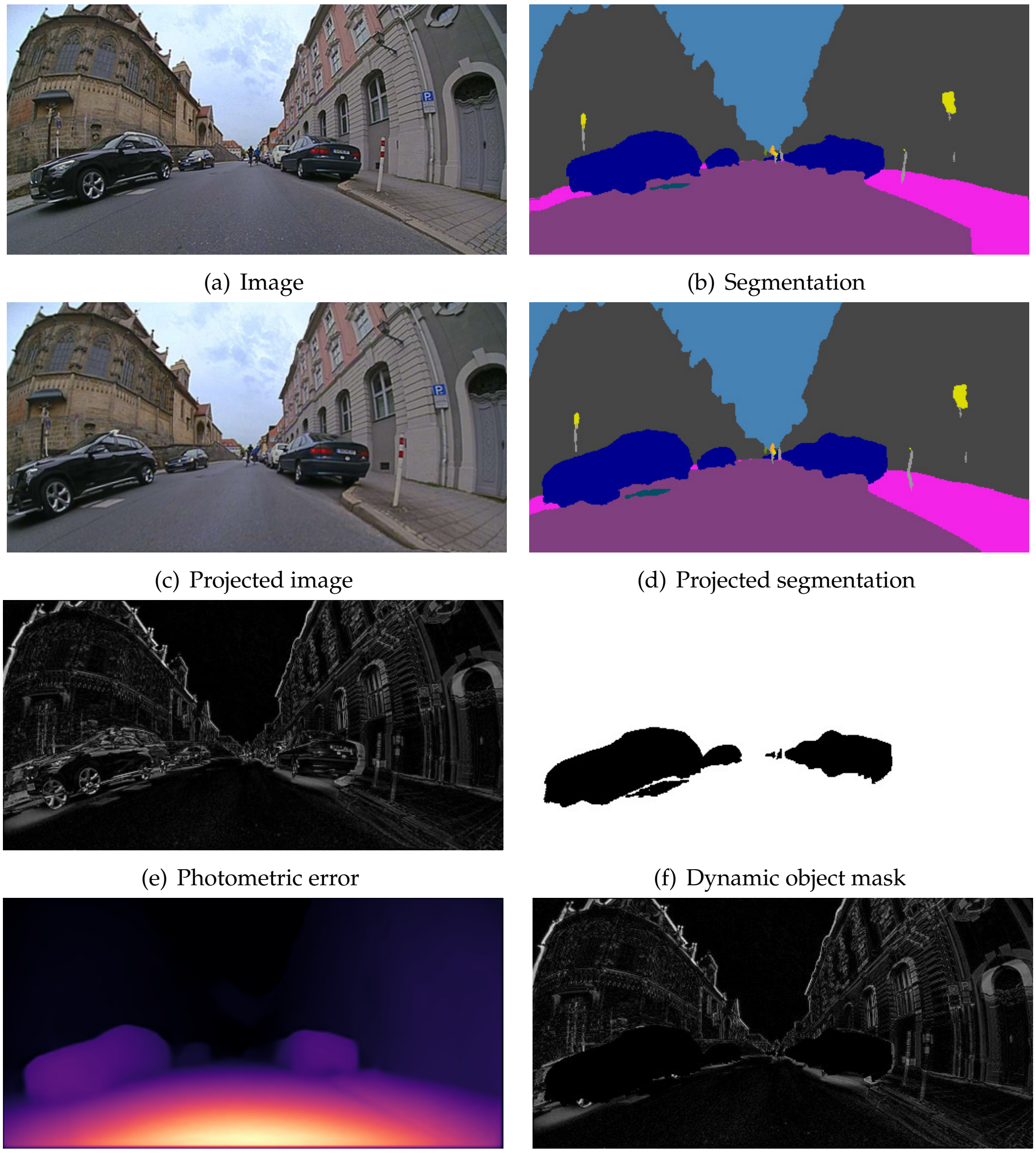
5.2.3 Dealing With Dynamic Objects

Typically, the assumed static world model for projections between image frames is violated by dynamic objects' appearance. Thereby, we use the segmentation masks to exclude *moving* potentially dynamic objects while *non-moving* dynamic object should still contribute.

In order to implement this, we aim at defining a pixel-wise mask μ_t, which contains a 0, if a pixel belongs to a dynamic object from the current frame I_t, or a wrongfully projected dynamic object from the reconstructed frames $\hat{I}_{t'\to t}$, and a 1 otherwise. For calculating the mask, we start by predicting a semantic segmentation mask M_t which corresponds to image I_t and segmentation masks $M_{t'}$ for all images $I_{t'}$. We then use the same projections for the images and warp the segmentation masks (using nearest neighbor instead of bilinear sampling), yielding projected segmentation masks $M_{t'\to t}$. Then, also defining the set of dynamic object classes $\mathcal{S}_{\text{DC}}\subset\mathcal{S}$ we can define μ_t by its pixel-wise elements at pixel location uv:

$$\mu_{t,uv}=\begin{cases}1, \; M_{t,uv}\notin\mathcal{S}_{\text{DC}}\;\wedge\; M_{t'\to t,uv}\notin\mathcal{S}_{\text{DC}}\\ 0, \; \text{else}\end{cases} \tag{5.7}$$

The mask is then applied pixel-wise on the reconstruction loss defined in Eq. 4.9 to mask out dynamic objects. However, as we only want to mask out *moving* DC-objects, we detect them using the consistency of the target segmentation mask and the projected segmentation mask to judge whether dynamic objects are moving between consecutive frames (*e.g.*, we intend to learn the distance of dynamic objects from


(a) Image

(b) Segmentation

(c) Projected image

(d) Projected segmentation

(e) Photometric error

(f) Dynamic object mask

(g) Distance Estimate

(h) Mask (f) applied on (e)

Figure 5.3 **Application of the semantic masking methods to handle potentially dynamic objects.** The dynamic objects inside the segmentation masks from consecutive frames in (b) and (d) are accumulated to a dynamic object mask, which is used to mask the photometric error (e), as shown in (h).

parking cars, but not from driving ones). With this measure, we apply the dynamic object mask μ_t only to an imposed fraction ϵ of images, in which the objects are detected as mostly moving.

5.2.4 Joint Optimization

We incorporate the task weighting approach by Kendall *et al.* [132]; we weigh the distance estimation and semantic segmentation loss terms for multi-task learning, which enforces homoscedastic (task) uncertainty. It is proven to be effective in weighing the losses from Eq. 4.14 and Eq. 5.1:

$$\frac{1}{2\sigma_1^2}\mathcal{L}_{tot} + \frac{1}{2\sigma_2^2}\mathcal{L}_{ce} + \log(1+\sigma_1) + \log(1+\sigma_2) \tag{5.8}$$

Homoscedastic uncertainty does not change with varying input data and is task-specific. We, therefore, learn this uncertainty and use it to downweigh each task. Increasing the noise parameter σ_i reduces the weight for the respective task. Furthermore, σ is a learnable parameter; the objective optimizes a more substantial uncertainty that should lead to a smaller contribution of the task's loss to the total loss. In this case, the different scales from the distance and semantic segmentation are weighed accordingly. The noise parameter σ_1 tied to distance estimation is relatively low compared to σ_2 of semantic segmentation, and the convergence occurs accordingly. Higher homoscedastic uncertainty leads to a lower impact on the task's network weight update. It is important to note that this technique is not limited to the joint learning of distance estimation and semantic segmentation but can also be applied to more tasks and arbitrary camera geometries.

5.2.5 Post-Processing Technique

This subsection provides a brief overview of how the distance maps get post-processed and converted to a representation directly used by motion planning. The prime purpose of surround-view cameras is to aid 360° near-field sensing around the car. Low-speed maneuvering such as parking requires very high accuracy in the order of 5 *cm* and the ability to detect small objects like curbs or potholes. We construct a 2D top-view heightmap grid at a 5 *cm* and 10 *m* range resolution by targeting these requirements. We project the distance and semantic segmentation maps from each image onto the top view and fill the height map cells with the height information. Due to a small overlap in the image's corners, we require a fusion scheme to combine the distance values. The spatial consistency of the scene across cameras can be exploited. We use a spatial smoothing filter to smoothen the current observation, further filtered using a temporal smoothing filter. The top-view post-processed distance and semantic maps are utilized for our **Level 3** planning module in the final step. The filtered height maps are illustrated in Figure 5.7. Additional qualitative results are illustrated in Figures 5.14, 5.15, 5.16 and 5.17.

5.3 Network Architecture

This section explains the novel architecture for semantically guided self-supervised distance estimation utilizing Camera Geometry Tensor (CGT) to handle multiple viewpoints and changes in the camera's intrinsic. The baseline from [2] used deformable convolutions to model the fisheye geometry to incorporate the distortion and improve the distance estimation accuracy. At first, we introduce a scalar-based self-attention encoder to obtain locally-attentive maps and a semantically guided decoder for the distance estimation using pixel-adaptive convolutions for the *SynDistNet* network architecture as shown in Figure 5.4, which can be trained in a one-stage fashion. Later, we incorporate improved vector-based self-attention modules from [8] shown in Figure 5.5. The figure provides an overview of the proposed network architecture used in the *SVDistNet* framework. The encoder is a self-attention network with pairwise and patchwise variants as described in Section 5.3.1. At the same time, the decoder uses pixel-adaptive convolutions, which are complemented by the novel

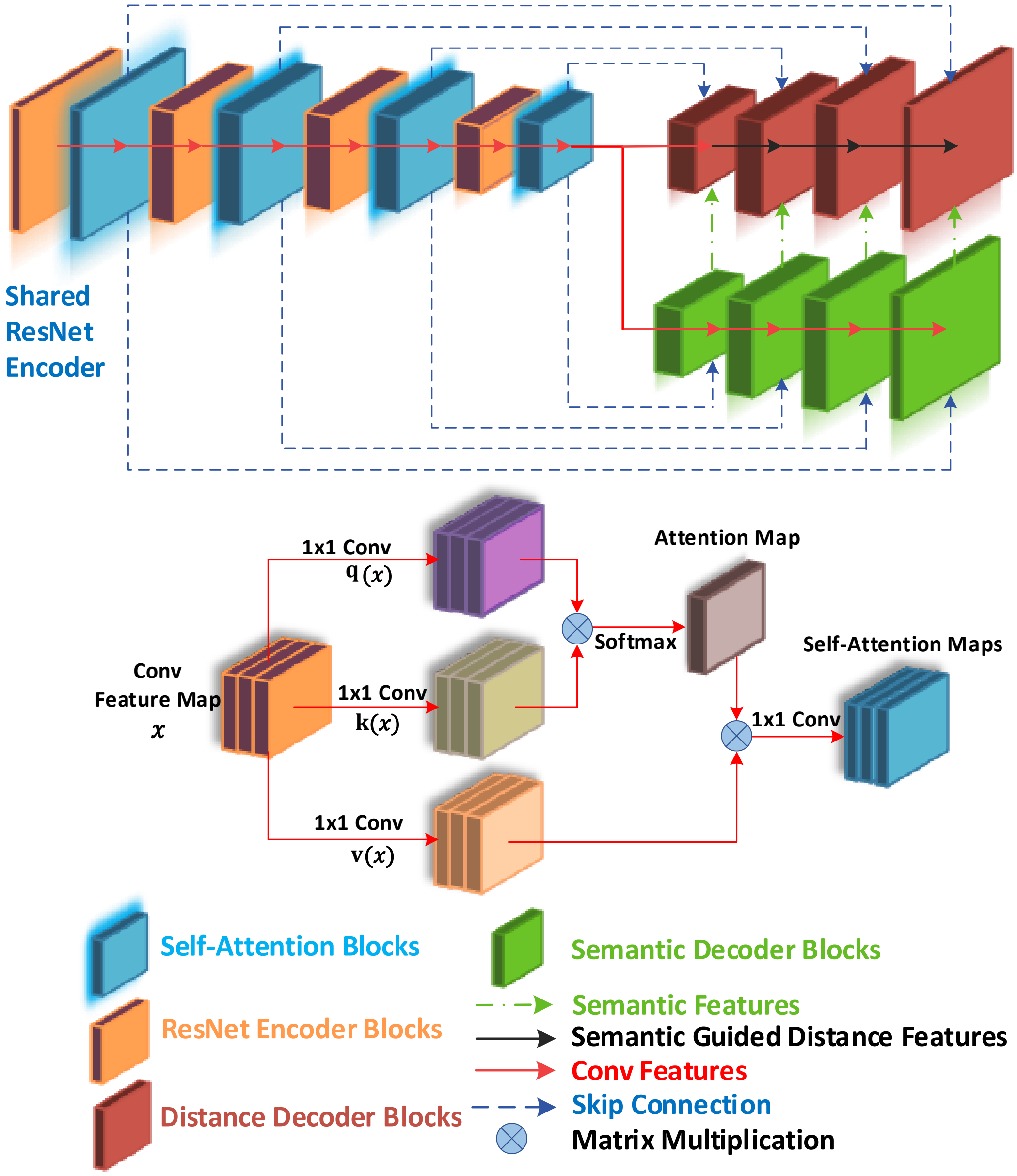


Figure 5.4 **Visualization of the SynDistNet network architecture.**

Camera Geometry Tensors. These networks efficiently adapt the weights across both spatial dimensions and channels. The complete training of both tasks is performed in a one-stage manner.

5.3.1 Self-Attention Encoder

Scalar based Self-Attention Encoder

Previous depth estimation networks [53, 55] use normal convolutions for capturing local information in an image, but the convolutions' receptive field is comparably small. One of the major drawbacks is that convolution lacks rotation invariance. The kernel K's footprint increases the number of parameters to be learned, and due to the filter's fixed nature, aggregation of neighborhood information can not adapt to its contents. Hu *et al.* [429] and Ramachandran *et al.* [430] perceived that self-attention could be a feasible choice for developing models for image perception rather than merely enhancing discrete convolutional operation layers. The authors present a self-attention layer which may replace convolution while reducing the number of parameters. Similar to a convolution, given a pixel $x_{uv} \in \mathbb{R}^{d_{in}}$ inside a feature map, the local region of pixels defined by positions $ab \in \mathcal{N}_k(uv)$ with spatial extent k centered around x_{uv} are extracted initially which is referred to as a memory block. For every memory block, the single-headed attention for computing the pixel output $z_{uv} \in \mathbb{R}^{d_{out}}$ is then calculated:

$$z_{uv} = \sum_{ab \in \mathcal{N}_k(uv)} \texttt{softmax}_{ab} \left(q_{ij}^{\top} k_{ab} \right) v_{ab} \tag{5.9}$$

where $q_{uv} = W_Q x_{uv}$ are the *queries*, *keys* $k_{ab} = W_K x_{ab}$, and *values* $v_{ab} = W_V x_{ab}$ are linear transformations of the pixel in position uv and the neighborhood pixels. The learned transformations are denoted by the matrices W. $\texttt{softmax}_{ab}$ defines a softmax applied to all logits computed in the neighborhood of uv. $W_Q, W_K, W_V \in \mathbb{R}^{d_{out} \times d_{in}}$ are trainable transformation weights. There exists an issue in the above-discussed approach, as there is no positional information encoded in the attention block. Thus the Eq. 5.9 is invariant to permutations of the individual pixels. For perception tasks, it is typically helpful to consider spatial information in the pixel domain. For example, the detection of a pedestrian is composed of spotting faces and legs in a proper relative localization. The main advantage of using self-attention layers in the encoder illustrated in Figure 5.4 is that it induces a synergy between geometric and semantic features for distance estimation and semantic segmentation tasks. In [431] sinusoidal embeddings are used to produce the absolute positional information. Following [430], instead of attention with 2D relative position embeddings, we incorporate relative attention due to their better accuracy for computer vision tasks. The relative distances of the position uv to every neighborhood pixel (a, b) is calculated to obtain the relative embeddings. The calculated distances are split up into row and column distances r_{a-i} and r_{b-j} and the embeddings are concatenated to form $r_{a-i,b-j}$ and multiplied by the query q_{uv}:

$$z_{uv} = \sum_{ab \in \mathcal{N}_k(ij)} \texttt{softmax}_{ab} \left(q_{ij}^{\top} k_{ab} + q_{ij}^{\top} r_{a-i,b-j} \right) v_{ab} \tag{5.10}$$

It ensures the weights calculated by the softmax function are modulated by both the key's relative distance and content from the query. Instead of focusing on the whole feature map, the attention layer only focuses on the memory block, which can be seen in the bottom part of Figure 5.4.

Vector based Self-Attention Encoders

Inspired by [8], we incorporate the self-attention network (SAN) backbone to the encoder and compare it with the standard ResNet18 [163] and ResNet50 [163] choice of encoder networks. Compared to the scalar attention self-attention layers explained earlier, which are only content-adaptive and not channel-adaptive. The vector attention work of Zhao [8] has a smaller footprint than the ResNet family of standard encoder heads, and the vector attention is both content and channel adaptive. The authors showcase two convolution variants, namely *pairwise* and *patchwise*, which may replace convolution while reducing the number of parameters and giving advantages in terms of robustness and generalization.

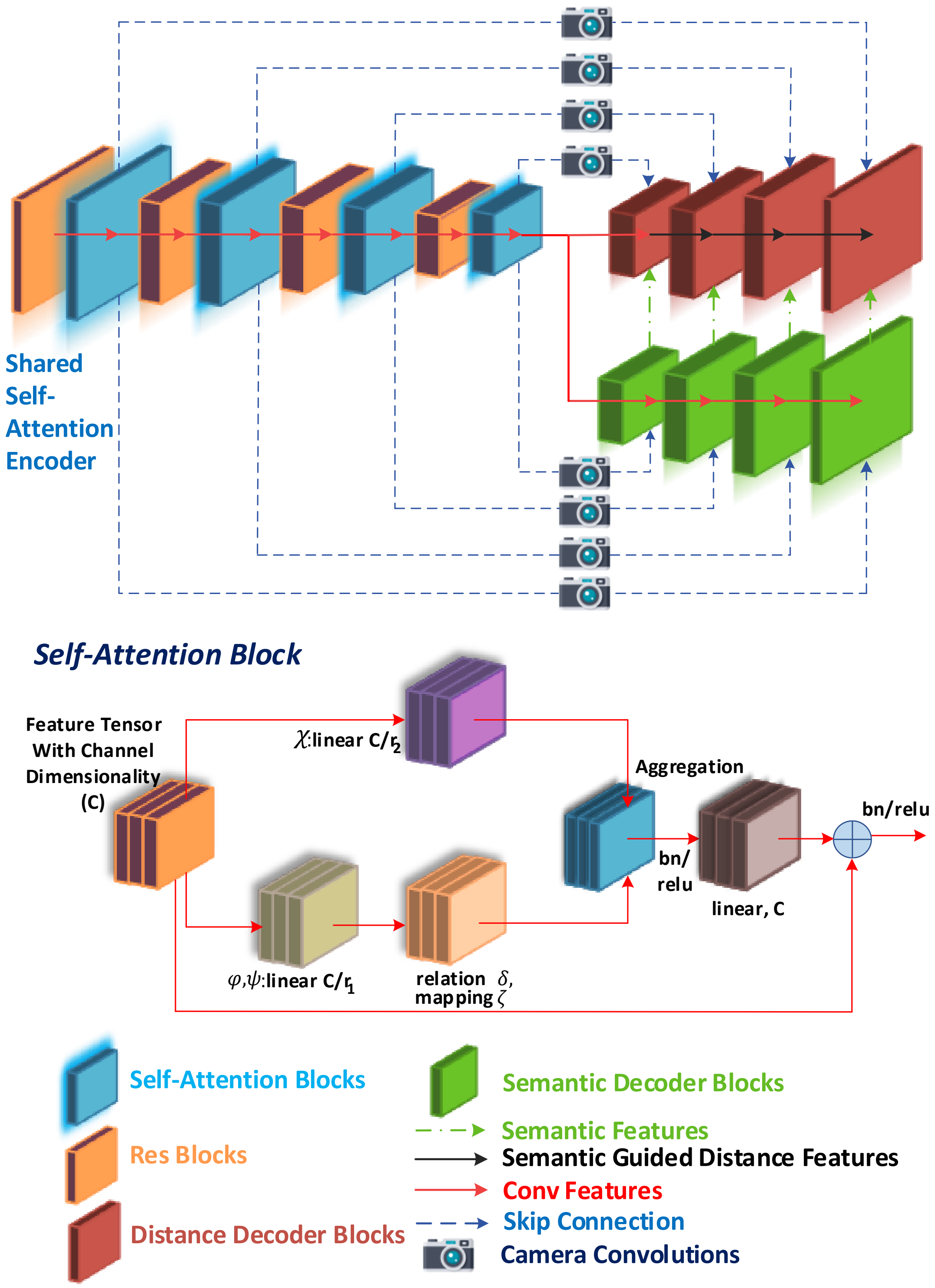


Figure 5.5 **Visualization of the SVDistNet network architecture.**

Pairwise Self-Attention

The *pairwise* self-attention module is given by:

$$z_{uv} = \sum_{ab \in \mathcal{N}_r(uv)} \eta(x_{uv}, x_{ab}) \odot \chi(x_{ab}) \tag{5.11}$$

The location of the spatial index in the feature vector x_{uv} is denoted by uv, $\odot$ is the Hadamard product, and the aggregation of the local footprint is $\mathcal{N}_r(uv)$. The new feature $z_{uv} \in \mathbb{R}^{d_{out}}$ is constructed by aggregating the feature vectors specified by the set of indices by the footprint $\mathcal{N}_r(uv)$. The adaptive weight vectors $\eta(x_{uv}, x_{ab})$ aggregate the feature vectors $\chi(x_{ab})$ produced by the function χ. The weights $\eta(x_{uv}, x_{ab})$ required to combine the transformed features $\chi(x_{ab})$ are computed by the function η. η is decomposed to elucidate the different forms of self-attention and is given by:

$$\eta(x_{uv}, x_{ab}) = \zeta(\delta(x_{uv}, x_{ab})) \tag{5.12}$$

The features x_{uv} and x_{ab} are expressed by a single vector outputted by the relation function δ. This vector is used along with the function ζ to map a vector that can be combined with $\chi(x_{ab})$ as shown in Eq. 5.11. The ζ function allows us to explore δ relationships that generate vectors of varying dimensionality that do not need to match the $\chi(x_{ab})$ dimensionality. In this work, we choose the relation function δ to be described by the *Hadamard product* form from [8]:

$$\delta(x_{uv}, x_{ab}) = \varphi(x_{uv}) \odot \psi(x_{ab}) \tag{5.13}$$

where φ and ψ are transformations that can be trained, which suit the dimensionality of the output. The dimensionality of $\delta(x_{uv}, x_{ab})$ is the same as that of the transformation functions with the Hadamard product.

Patchwise Self-Attention

The *patchwise* self-attention module is given by:

$$z_{uv} = \sum_{ab \in \mathcal{N}_r(uv)} \eta(x_{\mathcal{N}_r(uv)})_{ab} \odot \chi(x_{ab}) \tag{5.14}$$

In the footprint $\mathcal{N}_r(uv)$, the patch of feature vectors given by $x_{\mathcal{N}_r(uv)}$. $\eta(x_{\mathcal{N}_r(uv)})$ and the patch $x_{\mathcal{N}_r(uv)}$ have the same spatial dimensionality. $\eta(x_{\mathcal{N}_r(uv)})_{ab}$ is a vector located at ab in that tensor, corresponding to the x_{ab} vector in $x_{\mathcal{N}_r(uv)}$ spatially. The patchwise self-attention concerning the features x_{ab} is no longer a set operation compared to its pairwise counterpart. It is not permutation-invariant or cardinality-invariant: the $\eta(x_{\mathcal{N}_r(uv)})$ weight calculation will index the $x_{\mathcal{N}_r(uv)}$ feature vectors separately, by position, and can intermix information from feature vectors from various locations within the footprint. Therefore patchwise self-attention is especially more effective than standard convolution. $\eta(x_{\mathcal{N}_r(uv)})$ is decomposed by:

$$\eta(x_{\mathcal{N}_r(uv)}) = \zeta(\delta(x_{\mathcal{N}_r(uv)})). \tag{5.15}$$

The feature vector created by $\delta(x_{\mathcal{N}_r(uv)})$ is mapped by the function ζ to a tensor with the suitable dimensionality and it is comprised of weight vectors for all locations ab. Feature vectors x_{ab} from the patch $x_{\mathcal{N}_r(uv)}$ are combined by the function δ. We specifically incorporate the *Concatenation* form from [8] for the relation function δ.

$$\delta(x_{\mathcal{N}_r(uv)}) = [\varphi(x_{uv}), [\psi(x_{ab})]_{\forall ab \in \mathcal{N}_r(uv)}] \tag{5.16}$$

The pairwise and patchwise self-attention operations can be used to build residual blocks [163] for the encoder that performs both feature aggregation and transformation.

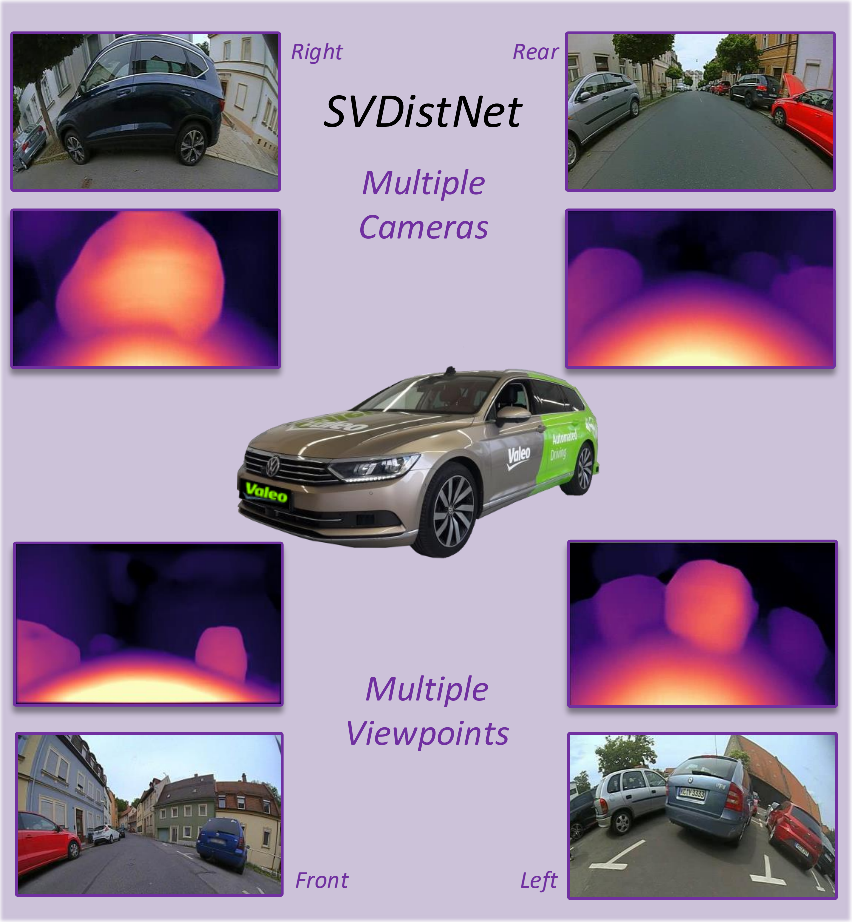


Figure 5.6 **Illustration of distance estimation on multiple cameras and multiple viewpoints constituting SVDistNet.**

5.3.2 Camera Geometry Tensor

Motivation

We target the design of a model that can be deployed in millions of vehicles having its own set of cameras. Although the underlying camera intrinsics model is the same for a particular family of vehicles, there are variations due to manufacturing processes, which require the calibration of each camera instance. Even after deployment, calibration can vary due to high environmental temperature or due to aging. Thus a calibration adaptation mechanism in the model is essential.

This contrasts with public datasets, which have a single camera instance for both the training and test dataset. In the Woodscape dataset, there are 12 different cameras with slight intrinsic variations to evaluate this effect. There are four camera instances around the vehicle with different intrinsics, even for a single instance of a surround-view system. A single model for these four cameras instead of 4 individual models would also have several practical advantages such as:

- An improved efficiency on the embedded system requiring less memory and data rate to transmit.
- An improved training by access to a larger dataset and regularization through different views.
- Maintenance and certification of a single model instead of four.

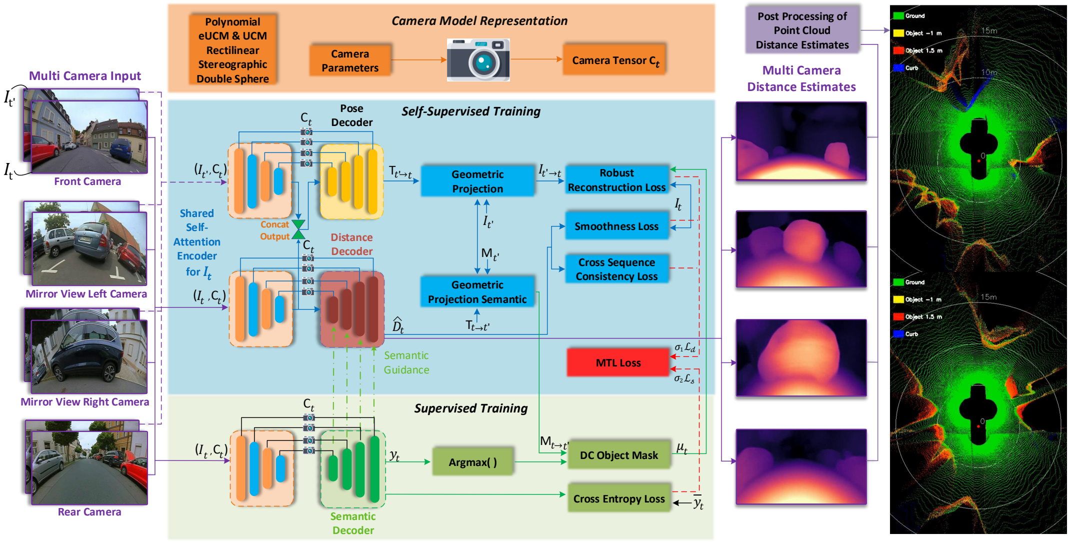


Figure 5.7 **Overview of SVDistNet: A surround-view based self-supervised distance estimation framework** making use of semantic guidance and camera-geometry adaptive convolutions (orange block).

Automated driving systems have a wide variety of cameras, typically around 10, placed in different car locations with different fields of view. Figure 5.6 shows sample distance estimation images of four cameras mounted on a car covering the entire 360° FoV surrounding the car. Instead of developing an individual model for each camera, developing a single model for all cameras is highly desirable, as discussed in the introduction. It is an unsolved problem, and we aim to solve this by incorporating camera geometry into distance estimation. We intend to convert all the camera geometry properties into a tensor called camera geometry tensor C_t (CGT), which will then be passed to the CNN model at both training and inference. From the view of distance estimation, camera intrinsics is the primary model adaptation needed. However, the CGT notion is generic, and we plan to extend it to include camera extrinsic and camera motion (visual odometry) for improving other tasks. The closest work is CAM-Convs [432], which uses camera-aware convolutions for pinhole cameras. We build upon this work and generalize to arbitrary camera geometries, including fisheye cameras.

The camera geometry adaptive mechanism is fundamental in the training process of the *SVDistNet* as the four different cameras mounted on the car have different intrinsic parameters and viewpoints. The trained distance and pose estimation networks need to generalize when deployed on a different car with a change in multiple viewpoints and intrinsics. To achieve this, we introduce the CGT in the mapping from RGB features to 3D information for distance estimation and semantic segmentation, as shown in Figure 5.5. We also add C_t to the pose encoder, shown in Figure 5.7. It is included in each stage and also applied to every skip connection. The framework consists of processing units to train the self-supervised distance estimation (blue blocks) and semantic segmentation (green blocks). The CGT (orange block) helps *SVDistNet* to yield distance maps on multiple camera-viewpoints and makes the network camera independent. C_t can also be adapted to the standard camera models, as explained in Section 5.3.2. Both modalities are weighted and optimized simultaneously by the multi-task loss from 5.8. The proposed framework can obtain top-view geometric information by post-processing the predicted distance and semantic maps in *3D space*.

The CGT is formulated in a three-step process: For efficient training, the pixel-wise coordinates and angle of incidence maps are pre-computed. The normalized coordinates per pixel are used for these channels by incorporating information from the camera calibration. We concatenate these tensors and represent them by C_t and pass it along with the input features to the SAN *pairwise* and *patchwise* operation modules. It comprises six channels in addition to the existing decoder channel inputs. The proposed approach can, in principle, be applied to any fisheye projection model of choice. We briefly discuss standard projection models which the CGT supports. For fisheye lenses, the mapping of 3D points to pixels universally requires a radial component $\varrho(\theta)$ [107]. The polynomial model is the commonly used one, and relatively recent projection models are UCM (Unified Camera Model) [433] and eUCM (Enhanced UCM) [82]. Rectilinear (representation of pinhole model) and Stereographic (mapping of a sphere to a plane) camera models are not suitable for fisheye lenses but provided for comparison. Double Sphere [83] is a recently proposed model with a closed-form inverse with low computational complexity. For further information on radial distortion models we refer Section 2.1.1.

The different maps included in the shared self-attention encoder are computed using the camera intrinsic parameters, where the distortion coefficients a_1, a_2, a_3, a_4 are used to create the angle of incidence maps (a_x, a_y), c_x, c_y are used to compute the principal point coordinate maps (cc_x, cc_y) and the camera's sensor dimensions (width w and height h) are utilized to formulate the normalized coordinate maps.

Centered Coordinates (cc_x, cc_y)

The information of the principal point position is fed to the SAN *pairwise* and *patchwise* operation modules by including the cc_x and cc_y coordinate channels centered at $(0, 0)$. We formulate cc_x and cc_y channels as shown below:

$$cc_x = \begin{pmatrix} 0 - c_x \\ 1 - c_x \\ \vdots \\ w - c_x \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}^{\mathsf{T}}_{(h+1)\times 1} = \begin{pmatrix} -c_x & \cdots & -c_x \\ \vdots & \ddots & \vdots \\ w - c_x & \cdots & w - c_x \end{pmatrix} \tag{5.17}$$

$$cc_y = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}_{(w+1)\times 1} \cdot \begin{pmatrix} 0 - c_y \\ 1 - c_y \\ \vdots \\ h - c_y \end{pmatrix}^{\mathsf{T}} = \begin{pmatrix} -c_y & \cdots & h - c_y \\ \vdots & \ddots & \vdots \\ -c_y & \cdots & h - c_y \end{pmatrix} \tag{5.18}$$

We concatenate cc_x and cc_y by resizing them using bilinear interpolation to match the input feature size.

Angle of Incidence Maps (a_x, a_y)

For the pinhole (Rectilinear) camera model, the horizontal and vertical angle of incidence maps are calculated from the cc maps using the camera's focal length f

$$a_{ch}[i, j] = \arctan\left(\frac{cc_{ch}[i, j]}{f}\right) \tag{5.19}$$

where ch can be x or y (see Eq. 5.17 and 5.18). For the different fisheye camera models, the angle of incidence maps can analogously be deduced by taking the inverse of the radial distortion functions $\varrho(\theta)$ listed above. Specifically, for the polynomial model used in this work, the angle of incidence θ is formulated by calculating the 4th order polynomial roots of $\varrho = \sqrt{x_I^2 + y_I^2} = a_1\theta + a_2\theta^2 + a_3\theta^3 + a_4\theta^4$ through a numerical method. We store the pre-calculated roots in a lookup table for all the pixel coordinates to achieve training efficiency and create the a_x and a_y maps by setting $x_I = cc_x[i,j], y_I = 0$ and $x_I = 0, y_I = cc_y[i,j]$ respectively. As UCM, eUCM, and Double Sphere projection models can be inverted analytically, there is no need for a lookup table.

Normalized Coordinates (nc_x, nc_y)

Additionally, we add two channels of normalized coordinates [432, 434] whose values vary between -1 and 1 linearly with respect to the image coordinates. The channels are independent of the camera sensor properties and characterize the spatial extent of the content in feature space in each direction (*e.g.*, a value of the $\hat{x}$ channel close to 1 indicates that the feature vector at this location is close to the right border).

5.3.3 Semantically-Guided Distance Decoder

To address the limitations of regular convolutions, we follow the approaches of [7, 435] in using pixel-adaptive convolutions (PAC) for semantic guidance inside the distance estimation branch of the multi-task network. This approach can break up the translation invariance of convolutions and incorporate spatially specific information of the semantic segmentation branch.

To this end, as shown in Figure 5.4 and Figure 5.5 we extract feature maps at different levels from the semantic segmentation branch of the multi-task network. These semantic feature maps are consequently used to guide the respective pixel-adaptive convolutional layer, following the formulation proposed in [435] to process an input signal x to be convolved:

$$x'_{uv} = \sum_{ab \in \mathcal{N}_k(i,j)} K(F_{uv}, F_{ab}) W[r_{a-i,b-j}] x_{ab} + B \tag{5.20}$$

where $\mathcal{N}_k(i,j)$ defines a $k \times k$ neighbourhood window around the pixel location uv (distance $r_{a-i,b-j}$ between pixel locations), which is used as input to the convolution with weights W (kernel size k), bias $B \in \mathbb{R}^1$ and kernel K, that is used in this case to calculate the correlation between the semantic guidance features $F \in \mathbb{R}^D$ from the segmentation network. We follow [7] in using a Gaussian kernel:

$$K(F_{uv}, F_{ab}) = \exp\left(-\frac{1}{2}(F_{uv} - F_{ab})^T \Sigma^{-1}_{uvab} (F_{uv} - F_{ab})\right) \tag{5.21}$$

with covariance matrix Σ_{uvab} between features F_{uv} and F_{ab}, which is chosen as a diagonal matrix $\sigma^2 \cdot \mathbf{1}^D$, where σ represents a learnable parameter for each convolutional filter.

In this work, we use pixel-adaptive convolutions to produce *semantic-aware distance features*. The fixed information encoded in the semantic network is used to disambiguate geometric representations for the generation of multi-level distance features.

Method	*Seg*	*Dist.*	*MTL*	*MTL (Synergy)*	*Seg. (mIoU)*	*Distance (RMSE)*
SynDistNet [4]	✓	✗	✗	✗	76.8	✗
	✗	✓	✗	✗	✗	2.316
	✓	✓	✓	✗	78.3	2.128
	✓	✓	✗	✓	**81.5**	**1.714**
SVDistNet (SAN10-patch)	✓	✗	✗	✗	77.1	✗
	✗	✓	✗	✗	✗	2.153
	✓	✓	✓	✗	78.6	1.861
	✓	✓	✗	✓	**82.3**	**1.532**

Table 5.1 **Effect of the multi-task training approaches SynDistNet and SVDistNet compared with each other.**

Compared to previous approaches [7, 189], we use features from the semantic segmentation branch that is trained simultaneously with the distance estimation branch introducing a more favorable one-stage training.

5.3.4 Comparison of Convolution vs. Self-Attention

The pairwise models match or outperform the convolutional baselines while requiring similar or less parameters and FLOP budgets. The patchwise models perform even better in terms of computational complexity. For example, the patchwise SAN10 with 11.8M params and 1.9G FLOPS outperforms ResNet50 with 25.6M params and 4.1G FLOPS, a 54% lower parameter and 44% lower FLOP count. SAN10-patch models' parameter count is almost nearly equivalent to ResNet18 with 11.7M params and 1.8G FLOPS, whereas SAN15-patch with 20.5M params and 3.3G FLOPS is equivalent to ResNet50's parameter count. The SAN10-pairwise with 10.5M params and 2.2G FLOPS outperforms ResNet18 with a 9% lower parameter count and 22% higher FLOP count. We could leverage the usage of a more robust loss function compared to L_1 to reduce training times on SAN10 by performing a single-scale image distance prediction in contrast to the multi-scale prediction in the previous works [2, 3].

5.4 Experiments

5.4.1 Ablative Experiments on Woodscape

Effect of Multi-Task Learning

Table 5.1 captures the primary goal of this work, which is to develop a synergistic multi-task network for semantic segmentation and distance estimation. ResNet-18 backbone was used for *SynDistNet* and SAN-10 for *SVDistNet*. We observe that both outputs improve through the multi-task training, mainly the distance estimation performance profits from the synergized semantic guidance. Firstly, we formulate single-task baselines for these tasks and build an essential shared encoder multi-task learning (MTL) baseline. The MTL results are slightly better than their respective single-task benchmarks demonstrating that shared encoder features can be learned for diverse tasks wherein segmentation captures semantic features and distance estimation captures geometric features. The proposed synergized MTL network

Network	*RL*	*Self-Attn*	*SEM*	*Mask*	*CGT*	Abs_{rel}	Sq_{rel}	RMSE	RMSE_{log}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
						lower is better				higher is better		
FisheyeDistanceNet [2]	✗	✗	✗	✗	✗	0.152	0.768	2.723	0.210	0.812	0.954	0.974
SynDistNet (ResNet18) [4]	✓	✗	✗	✗	✗	0.142	0.537	2.316	0.179	0.878	0.971	0.985
	✓	✗	✗	✓	✗	0.133	0.491	2.264	0.168	0.868	0.976	0.988
	✓	✓	✗	✓	✗	0.121	0.429	2.128	0.155	0.875	0.980	0.990
	✓	✓	✓	✗	✗	0.105	0.396	1.976	0.143	0.878	0.982	0.992
	✓	✓	✓	✓	✗	**0.076**	**0.368**	**1.714**	**0.127**	**0.891**	**0.988**	**0.994**
SynDistNet(ResNet50) [4]	✓	✗	✗	✗	✗	0.138	0.540	2.279	0.177	0.880	0.973	0.986
	✓	✗	✗	✓	✗	0.127	0.485	2.204	0.166	0.881	0.975	0.989
	✓	✓	✗	✓	✗	0.115	0.413	2.028	0.148	0.876	0.983	0.992
	✓	✓	✓	✗	✗	0.102	0.387	1.856	0.135	0.884	0.985	0.994
	✓	✓	✓	✓	✗	**0.068**	**0.352**	**1.668**	**0.121**	**0.895**	**0.990**	**0.996**
SVDistNet (SAN10-patch)	✓	✓	✗	✗	✗	0.128	0.469	2.153	0.164	0.875	0.974	0.986
	✓	✓	✗	✓	✗	0.114	0.413	2.022	0.149	0.878	0.982	0.990
	✓	✓	✗	✓	✓	0.101	0.378	1.861	0.133	0.884	0.984	0.991
	✓	✓	✓	✗	✗	0.094	0.345	1.789	0.128	0.887	0.985	0.992
	✓	✓	✓	✗	✓	0.082	0.316	1.682	0.119	0.890	0.987	0.993
	✓	✓	✓	✓	✗	0.074	0.343	1.641	0.112	0.892	0.985	0.994
	✓	✓	✓	✓	✓	**0.057**	**0.315**	**1.532**	**0.108**	**0.896**	**0.986**	**0.996**
SVDistNet (SAN10-pair)	✓	✓	✗	✓	✓	0.121	0.457	2.115	0.152	0.879	0.979	0.985
	✓	✓	✓	✗	✓	0.103	0.385	1.882	0.141	0.882	0.983	0.990
	✓	✓	✓	✓	✓	**0.081**	**0.365**	**1.710**	**0.128**	**0.890**	**0.985**	**0.994**
SVDistNet (SAN19-patch)	✓	✓	✗	✗	✗	0.121	0.437	2.127	0.153	0.878	0.976	0.989
	✓	✓	✗	✓	✗	0.109	0.408	2.006	0.145	0.880	0.982	0.992
	✓	✓	✗	✓	✓	0.098	0.372	1.849	0.138	0.884	0.983	0.991
	✓	✓	✓	✗	✗	0.091	0.351	1.773	0.129	0.886	0.986	0.993
	✓	✓	✓	✗	✓	0.070	0.305	1.669	0.108	0.893	0.986	0.994
	✓	✓	✓	✓	✗	0.067	0.296	1.578	0.106	0.895	0.985	0.994
	✓	✓	✓	✓	✓	**0.048**	**0.277**	**1.486**	**0.086**	**0.901**	**0.991**	**0.996**
SVDistNet (SAN19-pair)	✓	✓	✗	✓	✓	0.116	0.461	2.097	0.154	0.881	0.982	0.988
	✓	✓	✓	✗	✓	0.096	0.371	1.846	0.147	0.884	0.985	0.991
	✓	✓	✓	✓	✓	**0.074**	**0.331**	**1.624**	**0.101**	**0.891**	**0.986**	**0.994**

Table 5.2 **Ablation study on the effect of the contributions up to our final SVDistNet model on the Woodscape**. From our distance estimation baseline [2], we incrementally add up the robust loss (RL), self-attention layers encoder heads (Self-Attn), semantic guidance in the decoder (SEM), dynamic object masking (Mask), and camera geometry tensor (CGT).

SynDistNet reduces distance RMSE by 25% and improves segmentation accuracy by 4% as shown in Table 5.1. We break down these results further using extensive ablation experiments.

Later we compare the *SVDistnet* with *SynDistNet* on the Woodscape dataset. The MTL results and the single-task benchmark for distance estimation are significantly better than *SynDistNet* due to the usage of an improved self-attention encoder and the CGT. However, we observe only minimal gain for the semantic segmentation task. In the final experiment, we include the synergy between the distance and segmentation decoders. We observe that the content and channel adaptive self-attention encoder features can be learned jointly for these diverse tasks. The captured semantic features, used along with pixel-adaptive convolutions, guide the distance decoder to capture better geometric features. We break down these results further using ablation experiments.

At first, for the ablation analysis, we consider two variants of ResNet encoder heads. Distance estimation results of these variants are shown in Table 5.2. We showcase our improvements for various network architectures and, in particular, show the superiority of our *SVDistNet* model over the *SynDistNet* model as well as the positive effect of using the camera geometry tensor C_t. The distance estimates are capped to

40 m. We showcase the qualitative results of *SynDistNet* compared to the *FisheyeDistanceNet* model from **Chapter** 4 in Figure 5.10. Significant improvements in accuracy are obtained with the replacement of the L_1 loss by a generic parameterized loss function. The impact of the mask is incremental in the WoodScape dataset. Still, it poses the potential to solve the infinite depth/distance issue and provides a way to improve the photometric loss. With the proposed self-attention-based encoder coupled with the semantically-guided decoder architecture, we can consistently improve the performance. Finally, with all the additions, we outperform *FisheyeDistanceNet* for all considered metrics.

Additionally, we consider two variants of the self-attention encoder, namely pairwise and patchwise, as described in Section 5.3.1. We show the impact of specific contributions and their importance in the *SVDistNet* framework compared to the previous work *SynDistNet*. At first, we replace the L_1 loss with a generic parameterized loss function and test it using the self-attention encoder's patchwise variant, wherein we gain notable improvements in accuracy. We use the dynamic object mask obtained through projecting semantic segmentation predictions as described in Section 5.2.3 to filter all dynamic objects which contaminate the reconstruction loss. Additionally, this contribution possesses the potential to solve the infinite distance issue. We compare the baseline work *FisheyeDistanceNet* trained only for distance estimation to *SVDistNet*, which is trained in a multi-task fashion as shown in Figure 5.11 illustrate the semantic mask's impact on the estimates with additional results of *SVDistNet* in Figure 5.13. When adding the CGT to this setting, we observe a significant increase in accuracy since we train multiple cameras with different camera intrinsics and viewing angles. The aforementioned training strategy makes the network camera-independent and better generalizes images taken from a different camera.

To further improve the multi-task setup, we perform a synergy of distance and segmentation decoders. We provide semantic features from the supervised task to the distance decoder's geometric features, where we still train the distance estimation in a self-supervised fashion. In this setting, we drop the dynamic object mask and still achieve improvements.

We improve the metric results further by adding the CGT to this setting, and we could surpass the accuracy obtained by the best setting in *SynDistNet*. With the help of these vital features, we create an experiment comparable to *SynDistNet*'s final setting, which consolidates all the listed features, excluding the CGT. We could attain better results than *SynDistNet*'s ResNet50 results. It is important to note that ResNet50 is comparable to the SAN19-pair/patch encoder. However, we were able to outperform ResNet50 with the SAN10-patch encoder in terms of computational complexity (cf. Section 5.3.4).

We complete the *SVDistNet* model for the surround-view camera framework by introducing the CGT. In Figure 8.6, we show a few qualitative results of the failure cases having artifacts such as holes or merging of thin objects such as poles with the background. We also perform a few vital evaluations using the pairwise self-attention encoders. We were not able to obtain the same level of accuracy yielded by the patchwise self-attention encoder. The patchwise self-attention module is stringently more potent than the standard convolution, and the pairwise self-attention module matches or outperforms the convolutional equivalents. We perform the same vital ablation of the contributions with a higher-performing SAN19-patch encoder network.

Cams	*CGT*	Abs Rel	Sq Rel	RMSE	RMSE$_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
		lower is better				higher is better		
Front	✗	0.074	0.343	1.641	0.112	0.892	0.985	0.994
	✓	0.057	0.315	1.532	0.108	0.896	0.987	0.996
Rear	✗	0.089	0.358	1.657	0.131	0.888	0.981	0.988
	✓	0.065	0.337	1.579	0.123	0.891	0.983	0.992
Left	✗	0.102	0.398	1.874	0.126	0.886	0.980	0.983
	✓	0.091	0.382	1.781	0.114	0.889	0.985	0.990
Right	✗	0.105	0.406	1.889	0.135	0.882	0.980	0.981
	✓	0.093	0.391	1.796	0.120	0.887	0.983	0.986

Table 5.3 **Ablation study on multiple cameras** concerning the usage of Camera Geometry Tensor using the WoodScape.

Method	ρ	Abs Rel	Sq Rel	RMSE	RMSE$_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
		lower is better				higher is better		
FisheyeDistanceNet [2]	✗	0.152	0.768	2.723	0.210	0.812	0.954	0.974
SynDistNet	1	0.148	0.642	2.615	0.203	0.824	0.960	0.978
	0	0.151	0.638	2.601	0.205	0.822	0.962	0.981
	2	0.154	0.631	2.532	0.198	0.832	0.965	0.981
	$(0, 2)$	**0.142**	**0.537**	**2.316**	**0.179**	**0.878**	**0.971**	**0.985**
SVDistNet	2	0.145	0.564	2.279	0.186	0.863	0.966	0.982
	$0, 2$	**0.128**	**0.469**	**2.153**	**0.164**	**0.875**	**0.974**	**0.986**

Table 5.4 **Ablation study of the robust loss function of SynDistNet and SVDistNet on the WoodScape.**

Finally, with all the additions, we outperform all previous works [2, 3, 4] for all considered metrics on the Woodscape dataset.

Generalization Across Different Cameras

Table 5.3 depicts the generalization across different cameras from the surround-view setup using the CGT as described in Section 5.3.2. The qualitative results of the model are shown in Figure 5.8 where the same network is evaluated on four different fisheye cameras of a surround-view camera system. One can see that the *SVDistNet* model generalizes well across different viewing angles and consistently produces high-quality distance outputs. The metrics of each camera significantly improve as during the model's training phase, sequences from different cameras help in generalization. For example, the front camera's distance estimates profit as the side cameras steer the network to generalize close and overlapping objects. We test the *SVDistNet* model on an unseen sequence from one of the test cars whose cameras have different camera intrinsics than the ones used for training to examine the effect of the CGT as shown in Figure 5.9. Due to its usage, the network does not overfit to a particular camera intrinsic. It adapts to any changes from a family of unseen cameras deployed with a pre-calibrated camera setup. It leads to improved estimates and generalization of new cameras and allows training on images from different cameras.

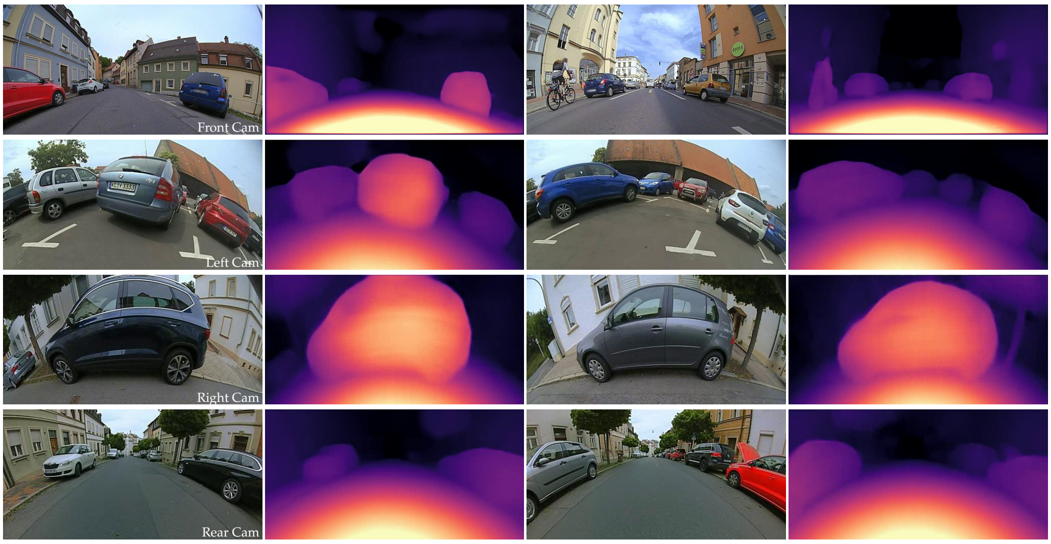


Figure 5.8 **Distance estimation results on a surround-view camera system.**

Robust loss function strategy

Table 5.4 ablates the usage of the robust loss function strategy on the frameworks *SynDistNet* and *SVDistNet*. We replace the L_1 loss with several variants of the general loss function varying the parameter α and observe a significant performance improvement. The L_1 loss is replaced with different variants of the robust general loss [179], and we showcase that the usage of adaptive or annealed variants of the loss can significantly improve the performance. The shape parameter ρ is varied, keeping the scale fixed with a general distribution than a fixed Laplacian distribution. Instead of an RGB representation, following [179], the YUV wavelet representations are used for the images, and the loss is applied to a YUV wavelet decomposition. The multi-scale training of the reconstruction loss described in Section 4.3.7 is dropped, which induces the sum of the loss means imposed at each scale in a D-level pyramid of side prediction since the robust loss function is a D level normalized wavelet decomposition. Compared to Barron *et al*. [179], we retain the edge smoothness loss described in Section 4.3.5 as it yielded better results. The fixed scale assumption is matched by setting the loss's scale c fixed to 0.01, which also roughly matches the shape of its L_1 loss. For the fixed scale models in Table 5.4, we used a constant value for ρ. We observe an improvement in the performance, and there is no single value of ρ, which is optimal. In the adaptive $\rho \in (0, 2)$ variant, ρ is made a free parameter and is allowed to be optimized along with the network weights during training. The adaptive plan of action outperforms the fixed strategies, which showcases the importance of allowing the model to regulate the robustness of its loss during training adaptively. A comparison of the adaptive model's performance with the fixed models indicates that no single set of α is optimal for all wavelet coefficients.

Online Refinement and Run-Time Comparison

In Table 5.5, we report the frame rate of all previous approaches and *SVDistNet* using different input resolutions and a 16-bit float precision ONNX model on NVIDIA's Jetson AGX platform, operating in full power mode. All the models, including the one with the highest resolution, are real-time capable and can be deployed in a car. The primary advantage of a single-frame distance estimator is its broad applicability.

Method	*Dataset*	*Network Resolution*	*Encoder head*	*Inference (fps)*
FisheyeDistanceNet [2]	K	640 x 192	ResNet-18	84
	K	1024 x 320		34
	WS	512 x 256		89
UnrectDepthNet [3]	K	640 x 192	ResNet-50	39
	K	1024 x 320		16
	WS	512 x 256		42
	WS	1024 x 512		11
SynDistNet [4]	K	640 x 192	ResNet-18	82
	K	1024 x 320		33
	WS	512 x 256		91
	WS	1024 x 512		23
	WS	512 x 256	ResNet50	42
SVDistNet	K	640 x 192	SAN-10	80
	K	1024 x 320		30
	WS	512 x 256		87
	WS	512 x 256	SAN-19	45

Table 5.5 **Ablation study on inference time (frames per second)** using ONNX 16-bit float precision models on NVIDIA's Jetson AGX on the KITTI (K) and the WoodScape (WS) datasets.

Method	Abs Rel	Sq Rel	RMSE	RMSE_{log}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
	lower is better				higher is better		
SAN10-patch	0.044	0.302	1.274	0.097	0.906	0.987	0.995
SAN10-pair	0.069	0.353	1.543	0.111	0.901	0.985	0.993
SAN19-patch	0.034	0.264	1.218	0.073	0.914	0.992	0.996
SAN19-pair	0.065	0.322	1.545	0.092	0.896	0.988	0.995

Table 5.6 **Online refinement** of the network's distance estimates incorporating [189], where the model is trained during inference on the WoodScape dataset.

Despite this, it comes at a cost when the continuous estimation of distances on image frames is misaligned or discontinuous, which is often the case. To overcome this issue, we follow the approach from [181, 189, 364], where we adapt the model in an online manner, mainly for practical autonomous systems. We train the model during inference by setting the batch size to 1. We feed in the inference image and its two adjacent frames, where we carry out the refinement as described in [189]. We do not implement any data augmentation techniques during this phase. With this technique, using a minimal temporal chain of frames (*i.e.*, three-frame snippets), the distance estimates enhance significantly, qualitatively, and quantitatively, as shown in Table 5.6. With a single frame's negligible delay, the *SVDistNet* framework can operate in real-time, even when using online refinement.

5.4.2 Ablative Experiments on KITTI

Pose Estimation Results

The PoseNet is an ego-motion predictor consisting of a SAN10-patch encoder. We apply the Siamese (twin network) notion where we feed I_t and I'_t individually to a shared self-attention encoder and concatenate the output features from the twin

Method	*No. of Frames*	*GT*	*Sequence 09*	*Sequence 10*
ORB-SLAM [108]	5	✓	0.014 ± 0.008	0.012 ± 0.011
DF-Net [416]	5	✓	0.017 ± 0.007	0.015 ± 0.009
SfMLearner [53]	5	✓	0.016 ± 0.009	0.013 ± 0.009
Klodt et al. [436]	5	✓	0.014 ± 0.007	0.013 ± 0.009
GeoNet [370]	5	✓	0.012 ± 0.007	0.012 ± 0.009
Struct2Depth [189]	5	✓	0.011 ± 0.006	0.011 ± 0.010
Ranjan [303]	5	✓	0.011 ± 0.006	0.011 ± 0.010
PackNet-SfM [7]	5	✓	0.010 ± 0.005	0.009 ± 0.008
PackNet-SfM [7]	5	✗	0.014 ± 0.007	0.012 ± 0.008
SVDistNet	5	✓	**0.009** ± **0.004**	0.008 ± **0.005**
SVDistNet	5	✗	0.010 ± 0.005	0.010 ± 0.008
DDVO [177]	3	✓	0.045 ± 0.108	0.033 ± 0.074
Vid2Depth [154]	3	✓	0.013 ± 0.010	0.012 ± 0.011
EPC++ [363]	3	✓	0.013 ± 0.007	0.012 ± 0.008
SVDistNet	3	✓	**0.011** ± **0.006**	**0.010** ± **0.007**
SVDistNet	3	✗	0.012 ± 0.007	0.011 ± 0.008
Monodepth2 [55]	2	✓	0.017 ± 0.008	0.015 ± 0.010
SVDistNet	2	✓	**0.015** ± **0.007**	**0.013** ± **0.007**
SVDistNet	2	✗	0.016 ± 0.008	0.014 ± 0.009

Table 5.7 **Evaluation of the pose estimation** on the KITTI Odometry Benchmark [116].

network before feeding it to the pose decoder as shown in Figure 5.7 predicting a relative pose between the images. Compared to the previous works [2, 3, 4], where we used Euler angles, we chose quaternions to represent the 3D rotation. The design choice is mainly due to its continuous and smooth representation of rotation and smaller memory footprint than rotation matrices. Also, quaternions are much more efficient than both matrix, and angle/axis representations used in [55, 181].

In Table 5.7, we report the average trajectory error (ATE) in meters, where we train the method on sequences 00-08 and evaluate on Sequences 09 and 10, same as for the baseline methods. For evaluation, we follow the evaluation protocol defined in [53]. Note that all the methods except the *SVDistNet* and PackNet-SfM utilize ground-truth at test-time to scale the prediction for a scale consistent result. We predict independent transformations for each of the four frame-to-frame transformations belonging to the five-frames set to evaluate the two-frame ego-motion model on the five-frame test sequences. We combine these different transformations to form local trajectories. We outperform the previous methods listed in Table 5.7, mainly by applying the bundle adjustment framework inflicted by the cross-sequence distance consistency loss [2]. It induces more constraints and simultaneously optimizes distances and camera pose for an implicitly extended training input sequence. This provides additional consistency constraints that are not induced by previous methods.

State-of-the-Art Comparison on KITTI

As there is little work on fisheye distance estimation, we evaluate the method on the extensively used KITTI dataset using the metrics illustrated in Section 2.2 by Eigen *et al.* [50]. The quantitative results are shown in the Table 5.8 illustrates that the

	Method	Resolution	Abs Rel	Sq Rel	RMSE	RMSE$_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
			lower is better				higher is better		
			KITTI						
Original [50]	EPC++ [363]	640 x 192	0.141	1.029	5.350	0.216	0.816	0.941	0.976
	Monodepth2 [55]	640 x 192	0.115	0.903	4.863	0.193	0.877	0.959	0.981
	PackNet-SfM [7]	640 x 192	0.111	0.829	4.788	0.199	0.864	0.954	0.980
	FisheyeDistanceNet [2]	640 x 192	0.117	0.867	4.739	0.190	0.869	0.960	0.982
	UnRectDepthNet [3]	640 x 192	**0.107**	0.721	4.564	**0.178**	0.894	0.971	**0.986**
	SynDistNet	640 x 192	0.109	**0.718**	**4.516**	0.180	**0.896**	**0.973**	**0.986**
	Monodepth2 [55]	1024 x 320	0.115	0.882	4.701	0.190	0.879	0.961	0.982
	FisheyeDistanceNet [2]	1024 x 320	0.109	0.788	4.669	0.185	0.889	0.964	0.982
	UnRectDepthNet [3]	1024 x 320	0.103	0.705	4.386	**0.164**	0.897	**0.980**	0.989
	SynDistNet	1024 x 320	**0.102**	**0.701**	**4.347**	0.166	**0.901**	**0.980**	**0.990**
Improved [143]	SfMLeaner [53]	416 x 128	0.176	1.532	6.129	0.244	0.758	0.921	0.971
	Vid2Depth [154]	416 x 128	0.134	0.983	5.501	0.203	0.827	0.944	0.981
	DDVO [177]	416 x 128	0.126	0.866	4.932	0.185	0.851	0.958	0.986
	EPC++ [363]	640 x 192	0.120	0.789	4.755	0.177	0.856	0.961	0.987
	Monodepth2 [55]	640 x 192	0.090	0.545	3.942	0.137	0.914	0.983	0.995
	PackNet-SfM [7]	640 x 192	0.078	0.420	3.485	0.121	**0.931**	0.986	**0.996**
	UnRectDepthNet [3]	640 x 192	0.081	0.414	3.412	0.117	0.926	0.987	**0.996**
	SynDistNet	640 x 192	**0.076**	**0.412**	**3.406**	**0.115**	**0.931**	**0.988**	**0.996**

Table 5.8 **Quantitative performance comparison of SynDistNet with other self-supervised monocular methods for depths up to 80 m for the KITTI.** *Original* uses raw depth maps as proposed by [50] for evaluation, and *Improved* uses annotated depth maps from [143].

improved scale-aware self-supervised approach outperforms all the state-of-the-art monocular approaches. Figure 5.12 provides qualitative results of *SynDistNet* on the KITTI test dataset for the segmentation and distance estimation tasks. More specifically, we improve the baseline *FisheyeDistanceNet* with the usage of a general and adaptive loss function [179] which is showcased in Table 5.4 and better architecture. Compared to PackNet-SfM [7], which presumably uses a superior architecture than the ResNet18, where they estimate scale-aware depths with their velocity supervision loss using the ground truth poses for supervision, we only rely on speed and time data captured from the vehicle odometry, which is easier to obtain. The approach can be easily transferred to the domain of aerial robotics as well. We could achieve higher accuracy than PackNet, which can be seen in Table 5.8. At test-time, all methods excluding *FisheyeDistanceNet*, PackNet-SfM, and *SynDistNet* scale the estimated depths using median ground-truth LiDAR depth.

For an extensive overview of the previous monocular methods' results, including the surround-view approach CGT tensor, we create Table 5.9. First, we train and evaluate the depth maps generated from the LiDAR point clouds, where Table 5.9 shows that with the use of the contributions, *we outperform all previous methods*. Using the online refinement method from [189], we obtain a significant improvement, while the results are still superior to previous methods. When training and evaluating the improved KITTI labels for depth estimation, we can show a significant improvement compared to previous approaches. For *SVDistNet*, we use the general camera tensor C_t as described in Section 5.3.2 in the model, wherein instead of the angle of incidence maps, we employ the maps generated using Eq. 5.19 for pinhole cameras. We showcase a comparison of *SVDistNet*'s estimates with leader-board algorithms in Figure 5.18.

	Method	Train	Abs Rel	Sq Rel	RMSE	RMSE$_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
			lower is better				higher is better		
Original [50]	SfMLearner [53]	M	0.208	1.768	6.958	0.283	0.678	0.885	0.957
	DNC [427]	M	0.182	1.481	6.501	0.267	0.725	0.906	0.963
	Vid2Depth [154]	M	0.163	1.240	6.220	0.250	0.762	0.916	0.968
	LEGO [437]	M	0.162	1.352	6.276	0.252	0.783	0.921	0.969
	Kumar [182]	M	0.211	1.979	6.154	0.263	0.731	0.897	0.959
	Wang *et al.* [438]	M	0.158	1.277	5.858	0.233	0.785	0.929	0.973
	GeoNet [370]	M	0.155	1.296	5.857	0.233	0.793	0.931	0.973
	Cycle-SfM [439]	M	0.162	1.349	5.847	0.239	0.784	0.925	0.969
	Li *et al.* [440]	M	0.150	1.127	5.564	0.229	0.823	0.936	0.974
	DDVO [177]	M	0.151	1.257	5.583	0.228	0.810	0.936	0.974
	DF-Net [416]	M	0.150	1.124	5.507	0.223	0.806	0.933	0.973
	GANVO [441]	M	0.150	1.141	5.448	0.216	0.808	0.939	0.975
	Bian [442]	M	0.137	1.089	5.439	0.217	0.830	0.942	0.975
	EPC++ [369]	M	0.141	1.029	5.350	0.216	0.816	0.941	0.976
	CC [303]	M	0.140	1.070	5.326	0.217	0.826	0.941	0.975
	Struct2Depth [189]	M	0.141	1.036	5.291	0.215	0.816	0.945	0.979
	LearnK [192]	M	0.128	0.959	5.230	0.212	0.845	0.947	0.976
	SIGNet [362]	M	0.133	0.905	5.181	0.208	0.825	0.947	0.981
	DualNet [186]	M	0.121	0.837	4.945	0.197	0.853	0.955	0.982
	OmegaNet [304]	M	0.126	0.835	4.937	0.199	0.844	0.953	0.982
	SuperDepth [443]	M	0.116	1.055	-	0.209	0.853	0.948	0.977
	Monodepth2 [55]	M	0.115	0.903	4.863	0.193	0.877	0.959	0.981
	PackNet-SfM [7]	M	0.111	0.829	4.788	0.199	0.864	0.954	0.980
	FisheyeDistanceNet [2]	M	0.117	0.867	4.739	0.190	0.869	0.960	0.982
	SGDepth [354]	M	0.113	0.880	4.695	0.192	0.884	0.961	0.981
	Patil *et al.* [444]	M	0.111	0.821	4.650	0.187	0.883	0.961	0.982
	UnRectDepthNet [3]	M	0.107	0.721	4.564	0.178	0.894	0.971	0.986
	SynDistNet [4]	M	0.109	0.718	4.516	0.180	0.896	0.973	0.986
	Shu *et al.* [181]	M	0.104	0.729	4.481	0.179	0.893	0.965	0.984
	SVDistNet	M	**0.102**	**0.706**	**4.459**	**0.172**	**0.908**	**0.974**	**0.986**
	Struct2Depth [189]	M*	0.109	0.825	4.750	0.187	0.874	0.958	0.983
	GLNet [364]	M*	0.099	0.796	4.743	0.186	0.884	0.955	0.979
	Shu *et al.* [181]	M*	0.088	0.712	4.137	**0.169**	0.915	0.965	0.982
	SVDistNet	M*	**0.086**	**0.701**	**4.118**	0.170	**0.919**	**0.976**	**0.985**
Improved [143]	SfMLearner [53]	M	0.176	1.532	6.129	0.244	0.758	0.921	0.971
	Vid2Depth [154]	M	0.134	0.983	5.501	0.203	0.827	0.944	0.981
	GeoNet [370]	M	0.132	0.994	5.240	0.193	0.883	0.953	0.985
	DDVO [177]	M	0.126	0.866	4.932	0.185	0.851	0.958	0.986
	EPC++ [369]	M	0.120	0.789	4.755	0.177	0.856	0.961	0.987
	Monodepth2 [55]	M	0.090	0.545	3.942	0.137	0.914	0.983	0.995
	PackNet-SfM [7]	M	0.078	0.420	3.485	0.121	0.931	0.986	0.996
	UnRectDepthNet [3]	M	0.081	0.414	3.412	0.117	0.926	0.987	0.996
	SynDistNet [4]	M	0.076	0.412	3.406	0.115	0.931	0.988	0.996
	SVDistNet	M	**0.071**	**0.405**	**3.345**	**0.106**	**0.934**	**0.988**	**0.996**
	SVDistNet	M*	**0.059**	**0.392**	**3.206**	**0.097**	**0.935**	**0.989**	**0.995**

Table 5.9 **Evaluation of the KITTI Eigen split** compared to most of the previous self-supervised monocular depth estimation methods. Following best practices, we cap depths at 80 m. We also evaluate using the *Original* depth maps generated from raw point clouds as proposed by [50] as well as *Improved* annotated depth maps as introduced by [143]. M indicates that the model is trained on monocular image sequences. M* indicates the online refinement technique [189], where the model is trained during inference. Note that while most approaches use median scaling w.r.t. the ground truth at test-time for a scale-consistent prediction, we do not need to use this scaling method.

5.5 Conclusion

Geometry and appearance are two crucial cues of scene understanding, *e.g.*, in automotive scenes. This chapter developed a multi-task learning model to estimate metric distance and semantic segmentation in a synergized manner. Specifically, we leverage the semantic segmentation of potentially moving objects to remove wrongful projected objects inside the view synthesis step. We also proposed a novel architecture to semantically guide the distance estimation trainable in a one-stage fashion and introduce the application of a robust loss function application. The primary focus was to develop the proposed model for less explored fisheye cameras based on the WoodScape dataset.

This chapter discussed distance estimation in detail, which is a challenging and vital problem for autonomous driving. We have solved it successfully by developing novel methodologies and synergized multi-task learning approach, which is crucial for scene understanding. We advanced the problem of multi-camera distance estimation for surround-view fisheye cameras. We introduced a novel camera model adaptation mechanism wherein camera parameters are transformed into a tensor and used within the CNN model. The specific camera model parameters are used during training and inference. Using this technique, we demonstrate training of a single distance estimation model for twelve different cameras with different extrinsic and intrinsic parameters and achieve strongly improved results than training a specialized model for each camera variant. We demonstrated the effect of each proposed contribution individually and obtained state-of-the-art results on both WoodScape and KITTI datasets for self-supervised distance estimation. In the next chapter, we look into localization using perception algorithms. *i.e.*, 2D object detection.

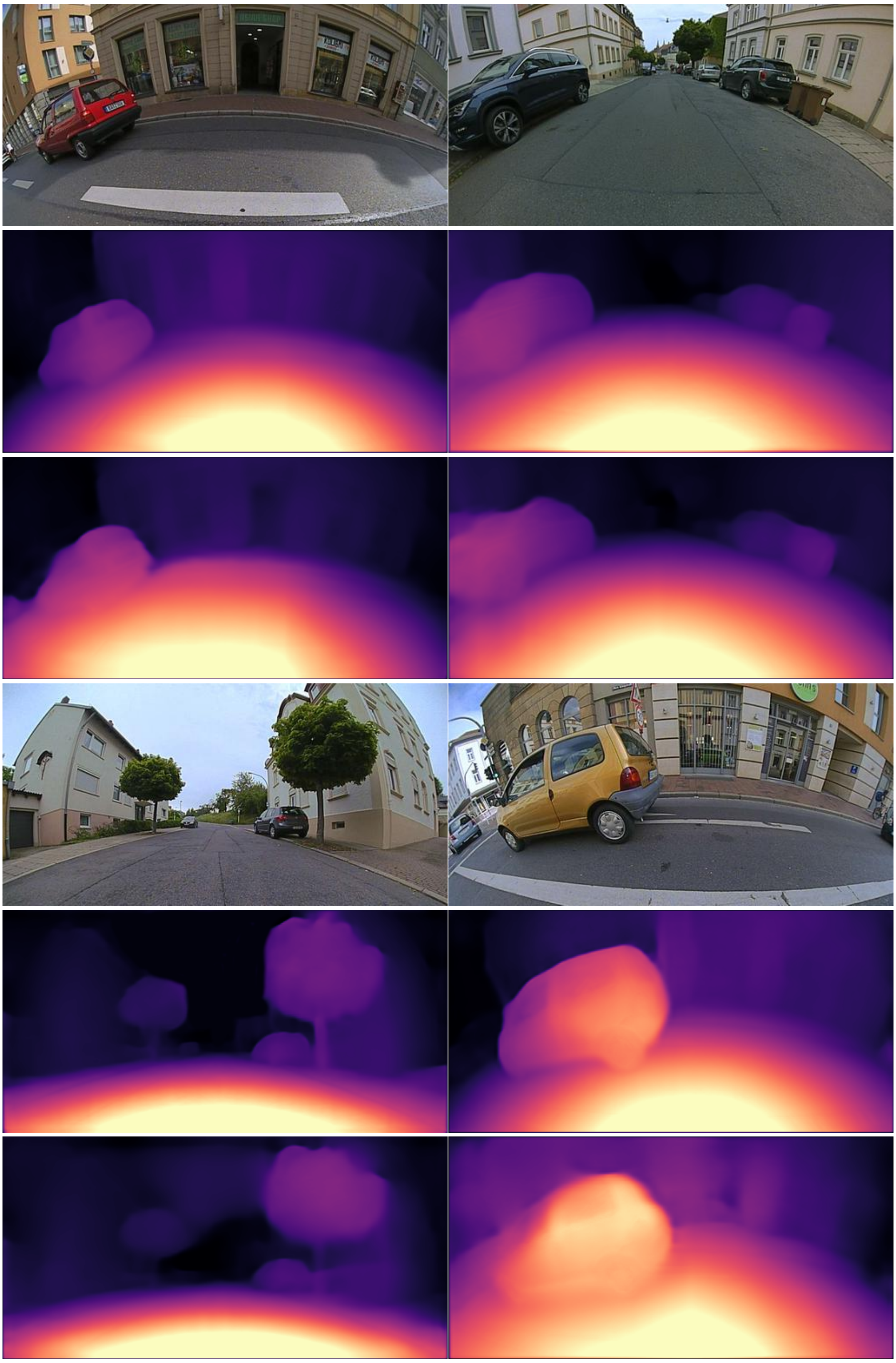


Figure 5.9 **Qualitative results of SVDistNet on an unseen sequences** from one of the test cars with a different camera intrinsic. The 1st and 4th row indicates the raw input images from the front and left camera. The 2nd, 5th, 3rd and 6th row indicate the distance estimates of the network trained with and without CGT respectively. Despite the notable variation in the camera parameters, the network with CGT outputs sharp distance maps on which edges are visible.

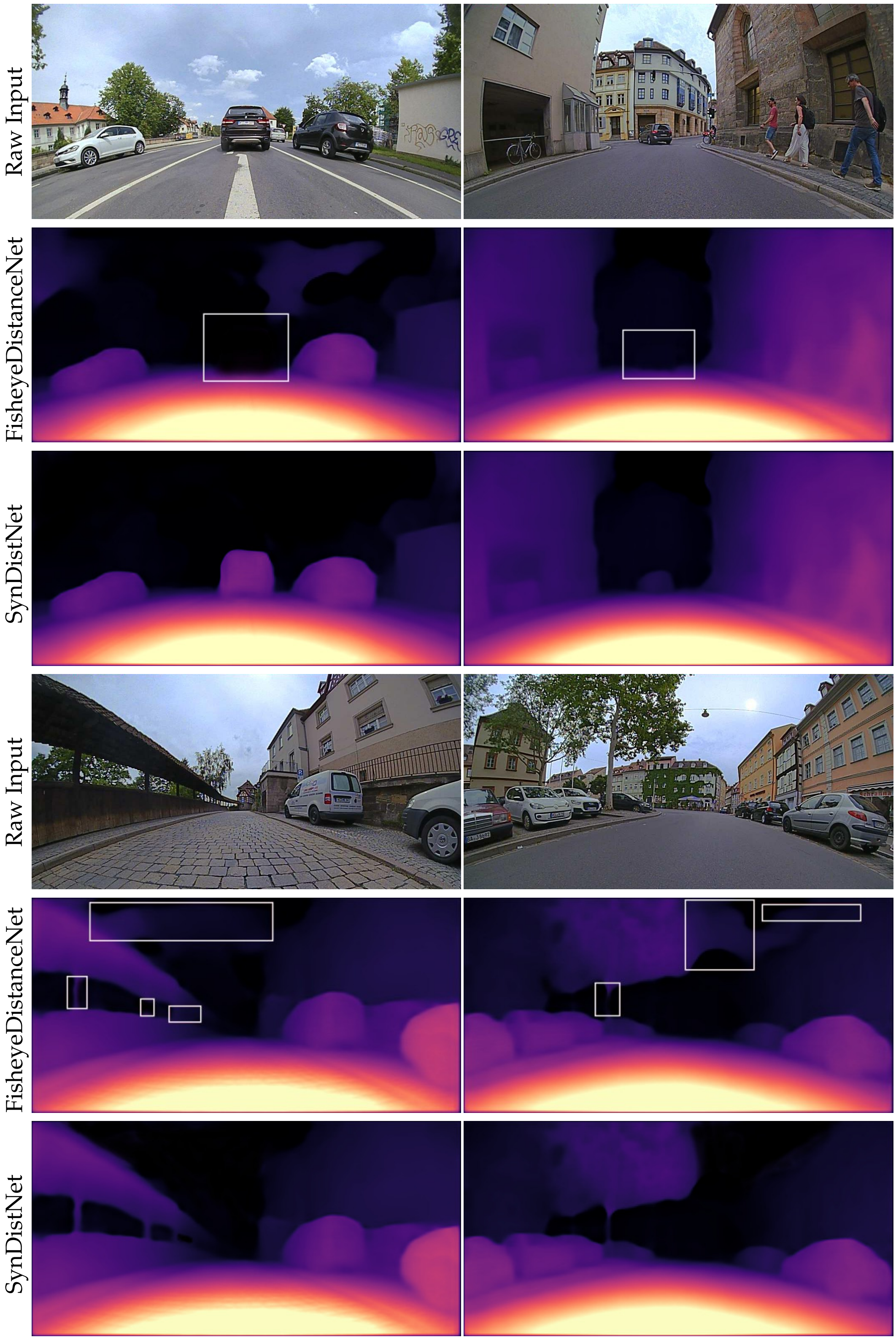


Figure 5.10 **Qualitative result comparison on the WoodScape** between the FisheyeDistanceNet and SynDistNet. SynDistNet can recover the distance of dynamic objects (left images), which eventually solves the infinite distance issue. In the 3rd and 4th rows, we can see that semantic guidance helps us to recover thin structures and resolves the distance of homogeneous areas, thereby outputting sharp distance maps on raw fisheye images.

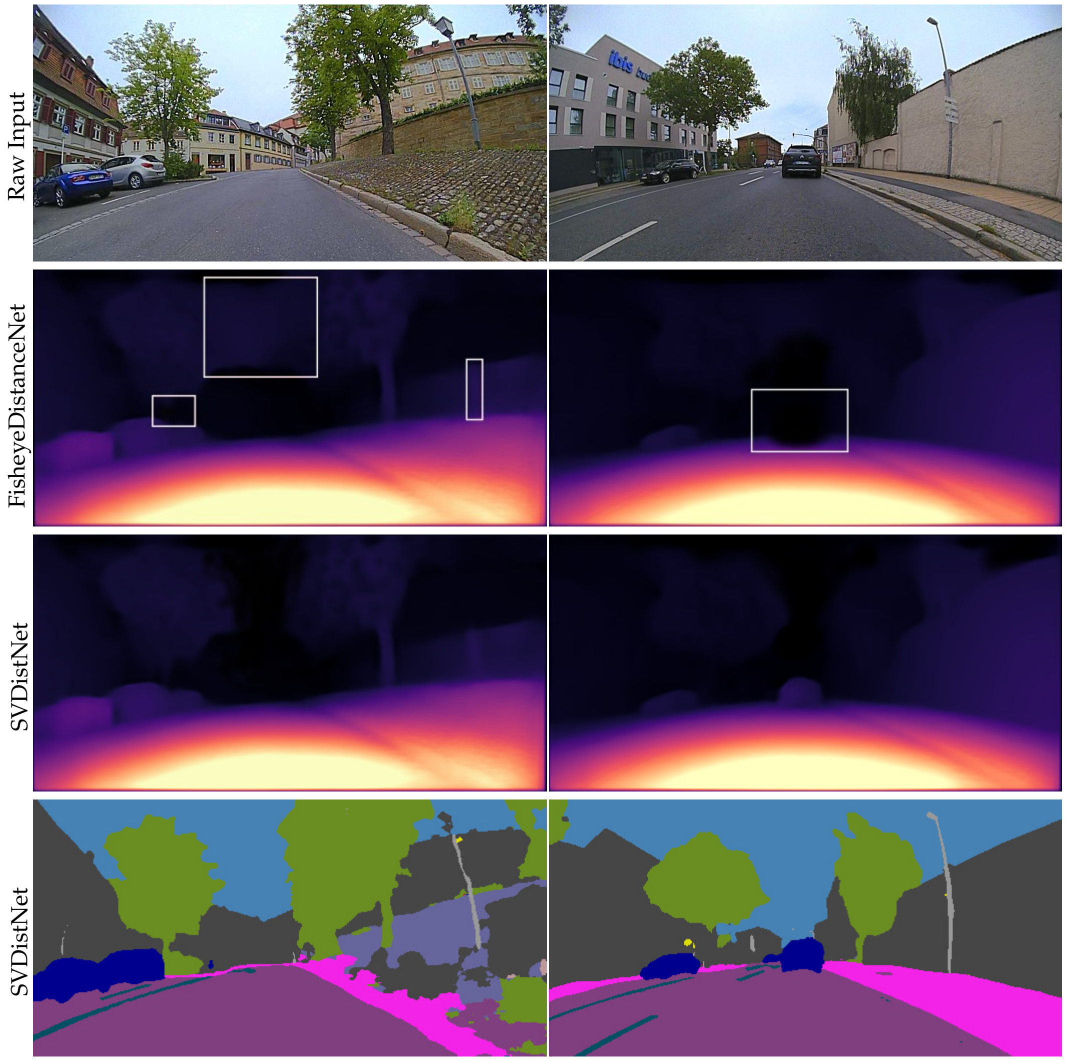


Figure 5.11 **Evaluation of the SVDistNet model on WoodScape.** We observe that using semantic guidance inside the SVDistNet model helps to recover thin structures inside the distance map (left images). We also solve the infinite distance issue for dynamic objects by incorporating the mask described in Section 5.2.3 (right images).

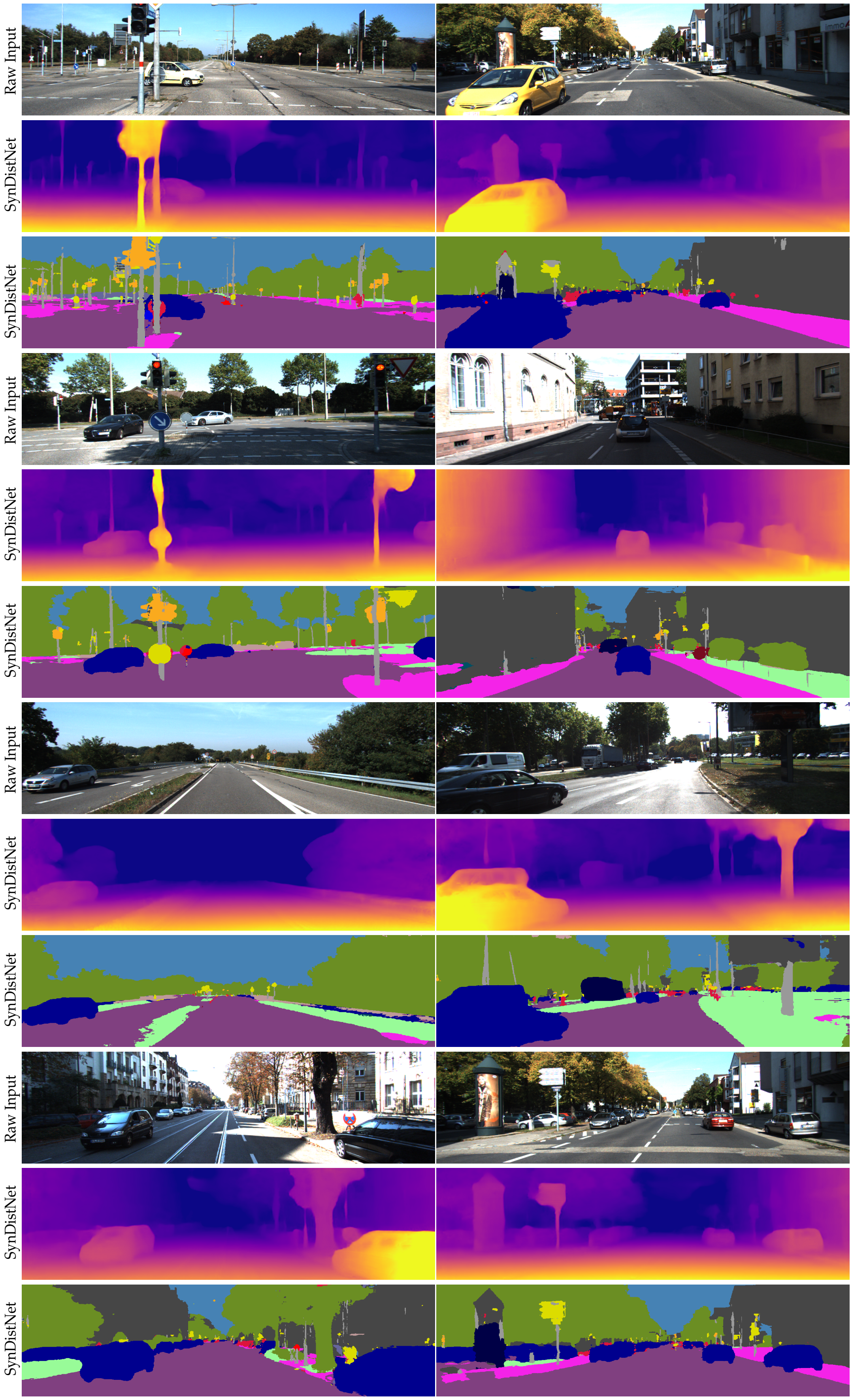


Figure 5.12 **Qualitative results of SynDistNet on the KITTI.** We showcase depth estimation as well as semantic segmentation outputs.

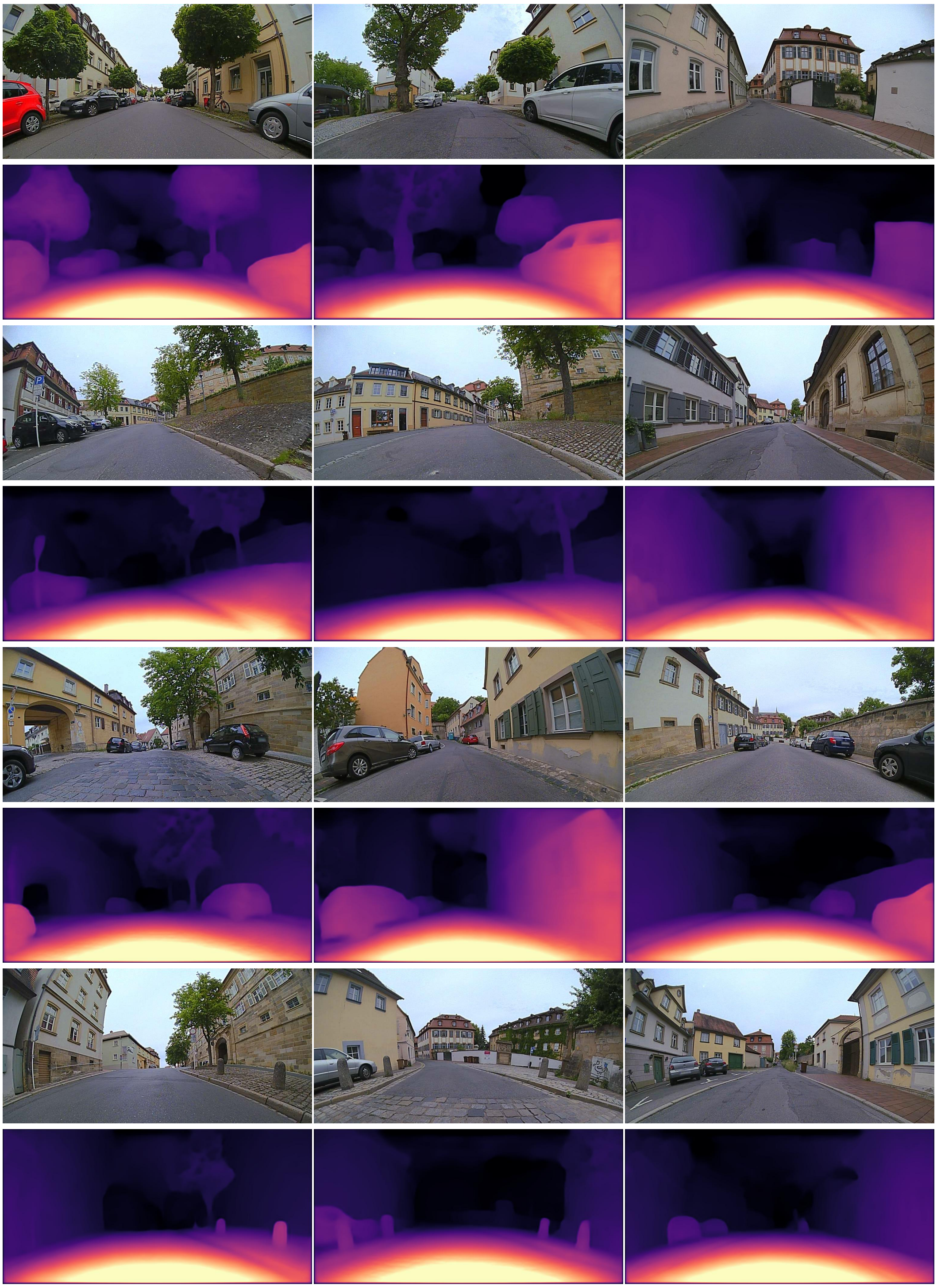


Figure 5.13 **Qualitative results of SVDistNet on WoodScape.** In the 4th row, we can see sharp curbs on the street, and in the 3rd row, we can see that the model adapts to the extreme distortion induced by the fisheye camera and produces sharp distance maps. Finally, the model adapts to the most complex scenes in the last few rows and produces very sharp scale-aware distance maps. For more qualitative results, we refer to this video: https://youtu.be/bmX0UcU9wtA.

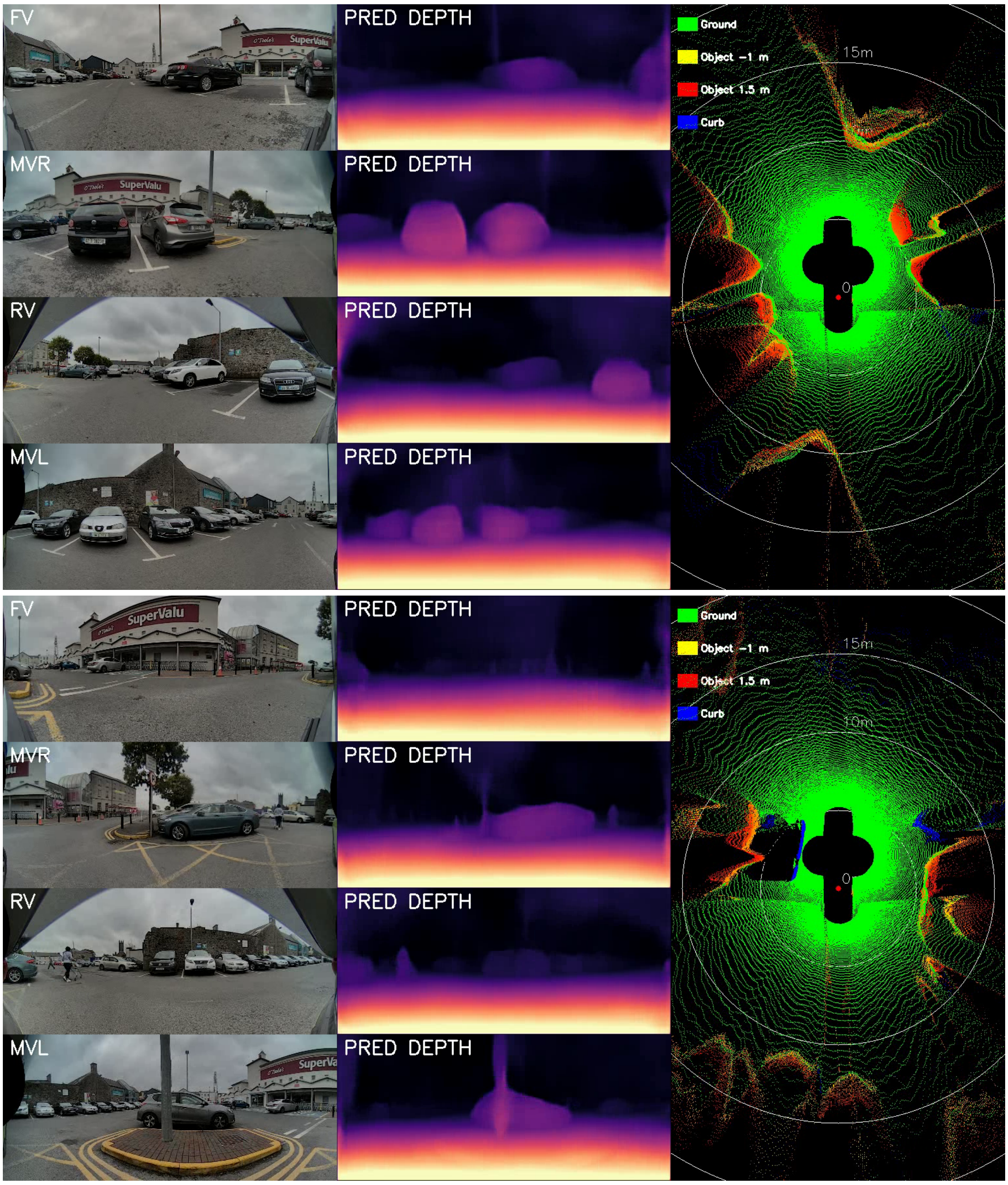


Figure 5.14 **Qualitative results of** 360° **bird's eye view distance output on an unseen sequence**. Two snapshots from a sequence illustrating color-coded heights and its corresponding surround-view images are shown. The distance maps are converted to color-coded height maps (*i.e.*, green is ground surface, yellow is an object at 1 m, red is an object at a distance of 1.5 m and blue indicates curb). Spatial and temporal smoothing operators were applied as discussed in subsection 5.2.5.

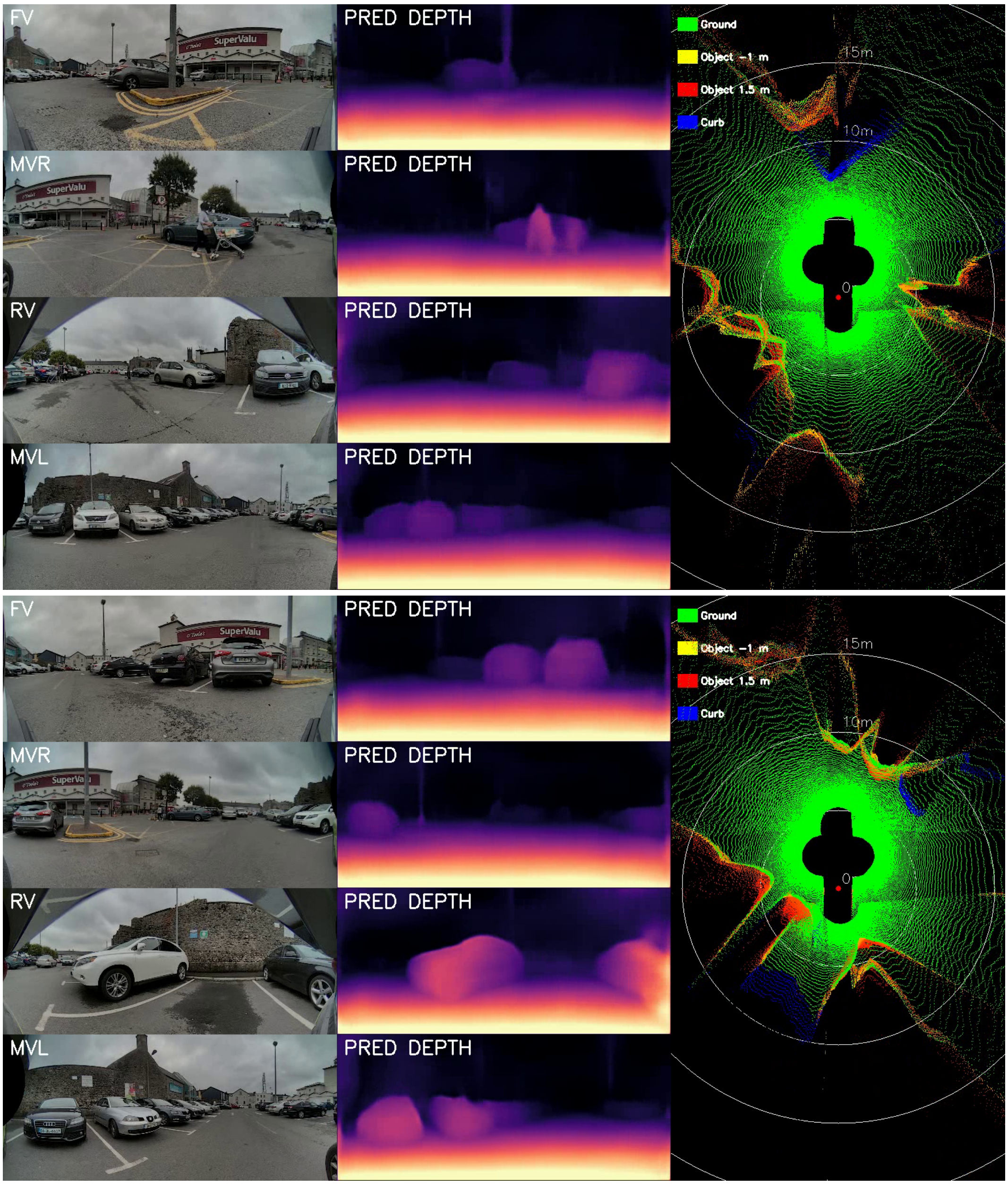


Figure 5.15 **Additional qualitative results of** 360° **bird's eye view distance output on an unseen sequence**. Two snapshots from a sequence illustrating color-coded heights and its corresponding surround-view images are shown. The distance maps are converted to color-coded height maps (*i.e.*, green is ground surface, yellow is an object at 1 m, red is an object at a distance of 1.5 m and blue indicates curb). Spatial and temporal smoothing operators were applied as discussed in subsection 5.2.5.

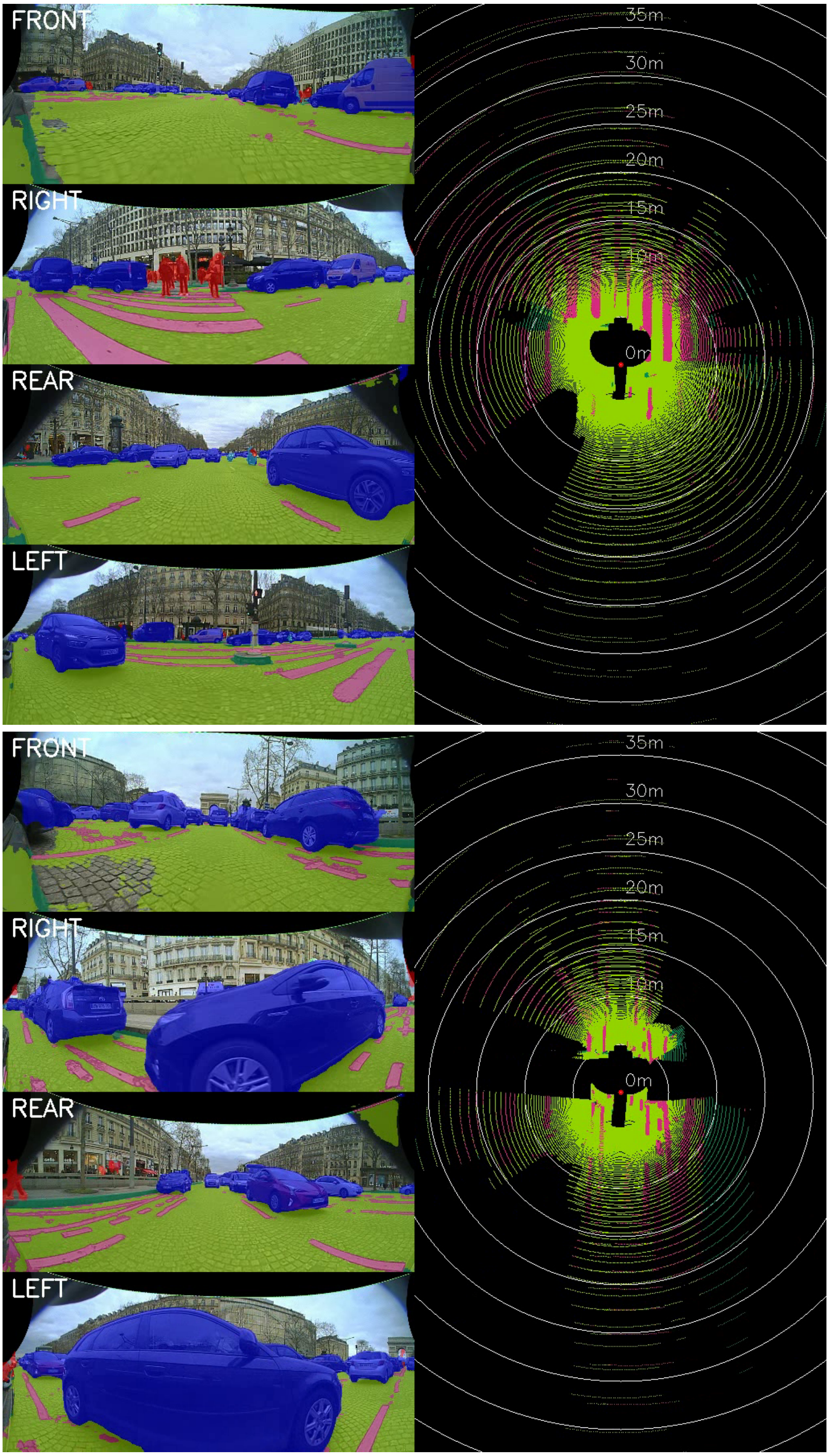


Figure 5.16 **Qualitative semantic segmentation results of** 360° **post-processed top-view output on an unseen sequence**. The semantic maps are converted to color-coded height maps (*i.e.*, green is the road surface, pink are the lane markings, dark green are the curbs). The top-view maps aid to detect free space for the autonomous vehicle to navigate.

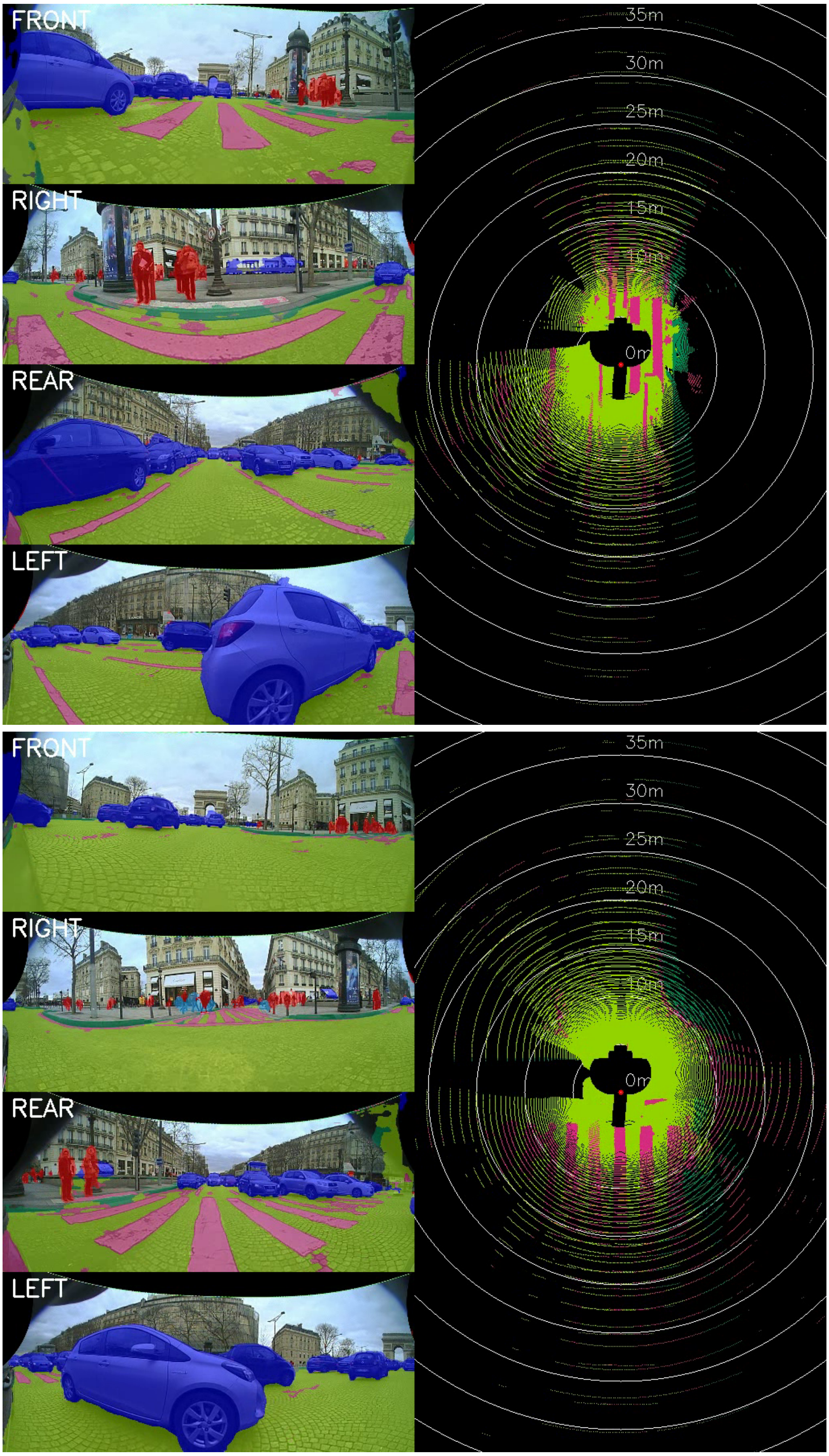


Figure 5.17 **Additional qualitative semantic segmentation results of** 360° **post-processed top-view outputs on an unseen sequence**. The semantic maps are converted to color-coded height maps (*i.e.*, green is road surface, pink is the lane marking, dark green is the curb). The top-view maps aid to detect free space for the autonomous vehicle to navigate.

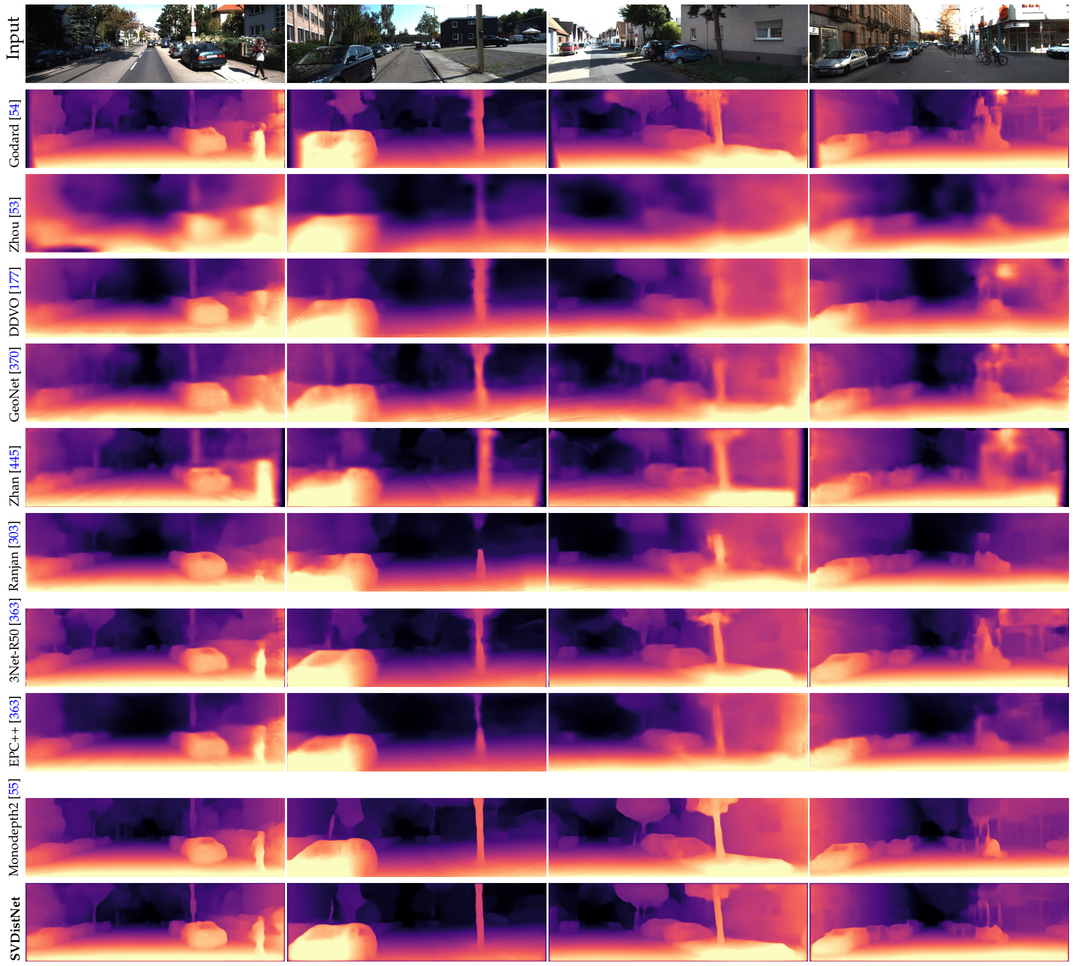


Figure 5.18 **Qualitative results of SVDistNet on KITTI compared with state-of-the-art algorithms.**

Chapter 6

Generalized Object Detection

Contents

6.1 Problem Definition

Object detection is a comprehensively studied problem in autonomous driving. However, it has been relatively less explored in the case of fisheye cameras. The standard bounding box fails in fisheye cameras due to the strong radial distortion, particularly in the image's periphery. We explore better representations such as oriented bounding box, ellipse, and generic polygon for object detection in fisheye images. We use the IoU metric to compare these representations using accurate instance segmentation ground truth. We design a novel curved bounding box model that has optimal properties for fisheye distortion models. In this thesis, to the best of our knowledge, we perform the first detailed study on object detection based on fisheye camera images for autonomous driving scenarios. To encourage further research, we also made a public release of a dataset of 10,000 images with annotations for all considered object representations. This work was formally presented as *Generalized Object Detection* [4] as an oral at the WACV conference in 2021. *My contribution to this work as a secondary*

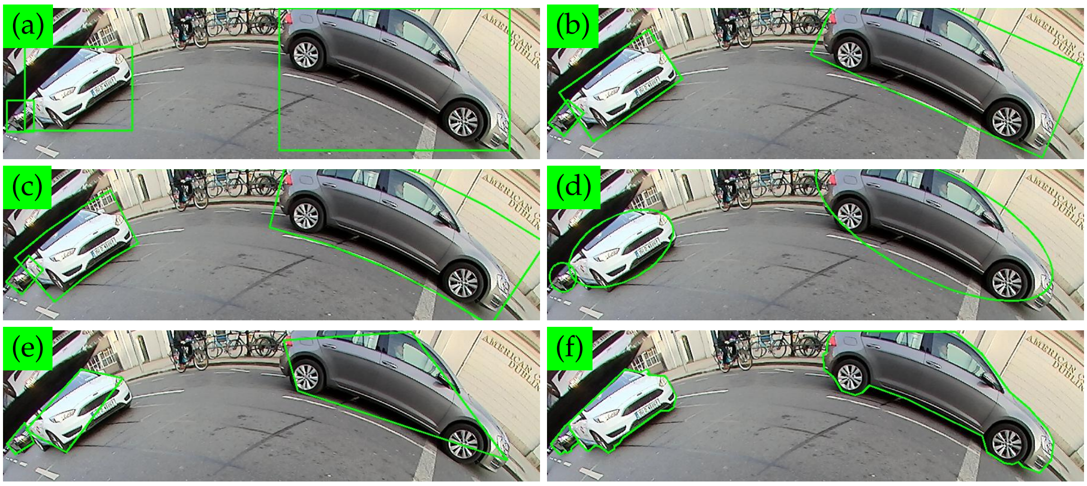


Figure 6.1 **Various 2D object detection representations on fisheye camera images.** (a) Standard Box, (b) Oriented Box, (c) Curved Box, (d) Ellipse, (e) 4-sided Polygon and (f) 24-sided Polygon.

author involved research idea discussion, writing code for (standard box model and training), code-reviews, generating a qualitative results video, and writing the research paper.

6.2 Object Representations

6.2.1 Adaptation of Box Representations

Standard Box Representation

The rectangular bounding box is the most common representation for object detection. They are aligned to the pixel grid axes, making them efficient to be regressed using a machine learning model. They are represented by four parameters ($\hat{x}$, $\hat{y}$, $\hat{w}$, $\hat{h}$), namely the box center, as well as width and height. It has the advantage of simplified, low-cost annotation. It also works in most cases, but it may capture a large non-object area within the box for complex shapes. This is particularly the case for distorted fisheye images, as shown in Figure 6.1 (a).

Oriented Box Representation

The oriented box is a simple extension of the standard box with an additional parameter $\hat{\theta}$ to capture the box's rotation angle. It is also referred to as a tilted or rotated box. Lienhart *et al*. [446] adapted the Viola-Jones object detection framework to output rotated boxes. It is also commonly used in LiDAR top-view object detection methods [116]. The orientation of the ground-truth range spans the range of (-90°to +90°), where this rotation angle is defined with respect to the x-axis. For this study, we used instance segmentation contours to estimate the optimally oriented box as a minimum enclosing rectangle.

Ellipse Representation

The ellipse representation is closely related to an oriented box and can be represented using the same parameter set. Width and height parameters represent the major and minor axis of the ellipse. In contrast to an oriented box, the ellipse has a smaller area at the edge, and thus it is better for representing overlapping objects, as shown for the objects at the very left in Figure 6.1. It may also help to fit some objects such as vehicles better than a box. We created the ground truth by fitting a minimum enclosing ellipse to the ground truth instance segmentation contours. In parallel work, Ellipse R-CNN [447] used ellipse representation for objects instead of boxes.

6.2.2 Distortion Aware Representation

This subsection aims to derive an optimal representation of objects undergoing radial distortion in fisheye images assuming a rectangular box is optimal for pinhole cameras. In the pinhole camera with no distortion, a straight line in the scene is imaged as a straight line in the image. However, a straight line in the scene is imaged as a curved segment in a fisheye image. The specific type of fisheye distortion determines the nature of the curved segment. The fisheye cameras from the Woodscape dataset we used are well represented and calibrated using a 4th order polynomial model for the fisheye distortion [12]. We are aware that there have been many developments in fisheye camera models over the past few decades, *e.g.* [80, 82, 448]. In this section, we consider the 4th order polynomial model and the division model only. The reason is that the 4th order polynomial model is provided by the dataset that we use, and we examine the division model to understand if the use of circular arcs is valid under such fisheye projections.

In this case, the projection of a line on to the image can be described parametrically with higher order polynomial curves. Let us consider a much simpler model for the moment - a first-order polynomial (or equidistant) model of a fisheye camera. *i.e.* $r' = a\theta$, where r' is the radius on the image plane, and θ is the angle of the incident ray against the optical axis. If we consider the parametric equation $\mathbf{P}(t)$ of a line in 3D Euclidean space:

$$\mathbf{P}(t) = \mathbf{D}t + \mathbf{Q} \tag{6.1}$$

where $\mathbf{D} = [D_x, D_y, D_z]$ is the direction vector of the line and $\mathbf{Q} = [Q_x, Q_y, Q_z]$ is a point through which the line passes. Hughes *et al.* [449] have shown that the projection on to a fisheye camera that adheres to equidistant distortion is described by:

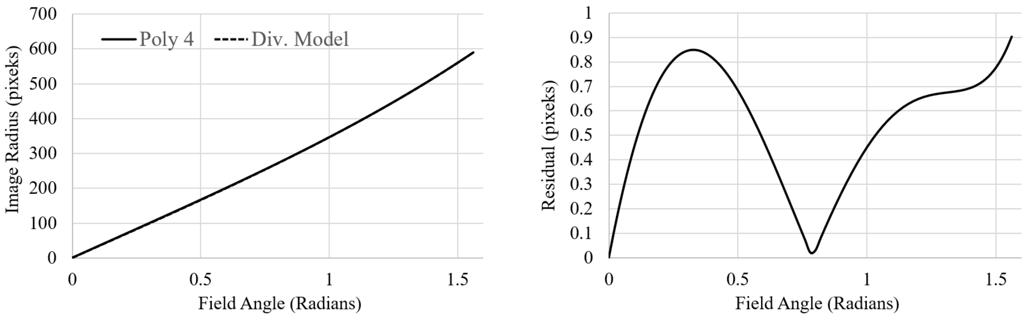
$$\mathbf{p}'(t) = \begin{bmatrix} D_x t + Q_x \\ D_y t + Q_y \end{bmatrix} \frac{|\mathbf{p}'(t)|}{|\mathbf{p}(t)|} \tag{6.2}$$

where

$$\frac{|\mathbf{p}'(t)|}{|\mathbf{p}(t)|} = \frac{a \arctan\left(\frac{d_{xy}(t)}{D_z t + Q_z}\right)}{d_{xy}(t)} \tag{6.3}$$

$$d_{xy}(t) = \sqrt{(D_x t + Q_x)^2 + (D_y t + Q_y)^2} \tag{6.4}$$

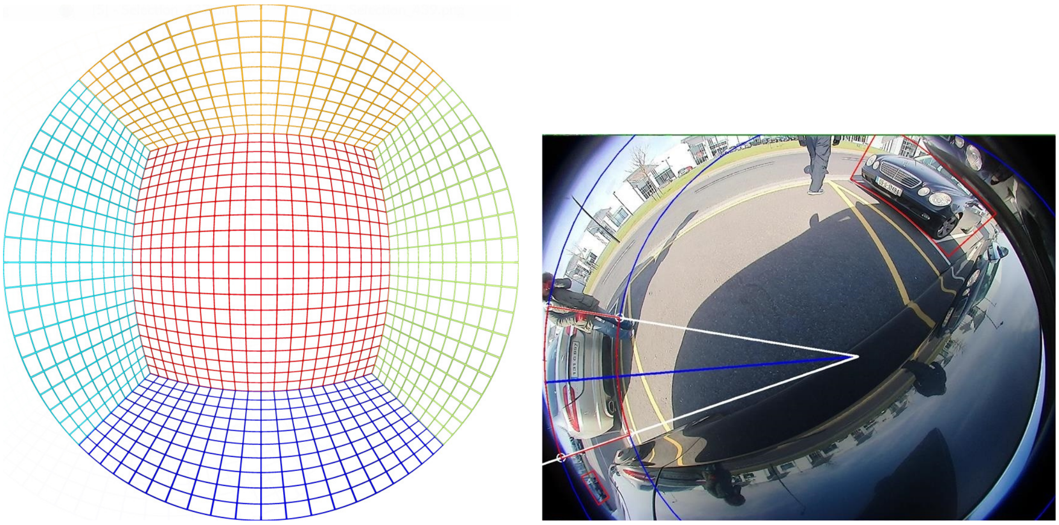
$\mathbf{p}(t)$ is the projected line in a pinhole camera, and $\mathbf{p}'(t)$ is the distorted image of the line in a fisheye camera. This is a complex description of a straight line's projection, especially considering that we have ignored all but the first-order polynomial term. Therefore, it is highly desirable to describe straight lines' projection using a more straightforward geometric shape. Bräuer-Burchardt and Voss [448] show that if the first-order *division model* can accurately describe the fisheye distortion, then we may use circles in the image to model the projected straight lines. As a note, the division



(a) Division model fit to the 4th order polynomial model. Note that the two are almost indistinguishable and looks overlayed.

(b) Residual error per field angle

Figure 6.2 **Approximation of the 4th order radial distortion model by the division model.**



(a) A 4th-degree polynomial model for radial distortion. We can visually notice that a box matures to a curved box, and it is justified theoretically in Section 6.2.2.

(b) **A Curved Bounding Box** captures the radial distortion and obtains a better footpoint. The center of the circle can be equivalently reparameterized using the object center $(\hat{x}, \hat{y})$.

Figure 6.3 **Illustration of fisheye distortion of projection of an Open Cube** and **Proposal of Curved Bounding Box** using a circle with an arbitrary center and radius.

model is generalized in [450], though it loses the property of straight line to circular arc projection. We should then consider how well the division model fits with the 4th order polynomial model. In [449], the authors adapt the division model slightly to include an additional scaling factor and prove that this does not impact the projection of a line to a circle. They show that the division model is a correct replacement for the equidistant fisheye model. Here we repeat this test but compare the division model to the 4th order polynomial. The results are shown in Figure 6.2. As can be seen, the division model can map to the 4th order polynomial with a maximum of < 1 pixel error. While this may not be accurate enough for applications in which sub-pixel error accuracy is desirable, it is sufficient for bounding box accuracy.

Therefore, we propose a novel curved bounding box representation using circular arcs. Figure 6.3 (left) provides a visual justification of circular arcs. We illustrate an open cube projection with grid lines where the straight lines become circular arcs

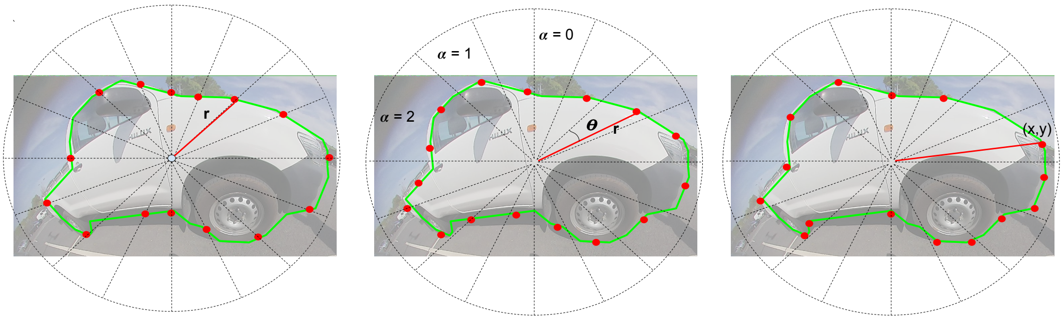


Figure 6.4 **Generic Polygon Representations. Left:** Uniform angular sampling where the intersection of the polygon with the radial line is represented by one parameter per point (r). **Middle:** Uniform contour sampling using L2 distance. It can be parameterized in polar co-ordinates using 3 parameters (r, θ, α). α denotes the number of polygon vertices within the sector, and it may be used to simplify the training. Alternatively, two parameters (x,y) can be used, as shown in the figure on the right. **Right:** Variable step contour sampling. It is shown that the straight line in the bottom has fewer points than curved points, such as the wheel. This representation allows to maximize the utilization of vertices according to local curvature.

after projection. Figure 6.3 (right) illustrates the details of the curved bounding box. The blue line represents the axis, and the white lines intersect with the circles creating starting and ending points of the polygon. This representation allows two sides of the box to be curved, giving the flexibility to adapt to image distortion in fisheye cameras. It can also specialize in an oriented bounding box when there is no distortion for the objects near the principal point.

We create an automatic process to generate the representation that takes an object contour as an input. First, we generate an oriented box from the output contour. We choose a point that lies on the oriented box's axis line to represent a circle center. From the center, we create two circles intersecting with the corner points of the bounding box. We construct the polygon based on the two circles and the intersection points. To find the best circle center, we iterate over the axis line and choose the circle center, which forms a polygon with the minimum IoU with the instance mask. The output polygon can be represented by 6 parameters, namely, $(c_1, c_2, r_1, r_2, \theta_1, \theta_2)$ representing the circle center, two radii and angles of the start and end points of the polygon relative to the horizontal x-axis. By simple algebraic manipulation, we can re-parameterize the curved box using the object center $(\hat{x}, \hat{y})$ following a typical box representation instead of the center of the circle.

6.2.3 Generic Polygon Representations

The polygon is a generic representation for an arbitrary shape and is typically used even, for instance segmentation annotation. Thus a polygon output can be seen as a coarse segmentation. We discuss two standard representations of a polygon and propose a novel extension that improves accuracy.

Uniform Angular Sampling

The polar representation is quite similar to the PolarMask [451], and PolyYOLO [452] approaches. As illustrated in Figure 6.4 (left), the full angle range of 360° is split into N equal parts where N is the number of polygon vertices. Each polygon vertex is represented by the radial distance r from the centroid of the object. Uniform angular sampling removes the need for encoding the θ parameter. The polygon is finally represented by its object center $(\hat{x}, \hat{y})$ and radius $\{r_i\}$.

Uniform Perimeter Sampling

In this representation, we divide the perimeter of the object contour equally to create N vertices. Thus the polygon is represented by a set of vertices $\{(x_i, y_i)\}$ using the centroid of the object as the origin. PolyYOLO [452] showed that it is better to learn polar representation of the vertices $\{(r_i, \theta_i)\}$ instead. They define a parameter α to denote the presence or absence of a vertex in a sector, as shown in Figure 6.4 (middle). We extend this parameter to be the count of vertices in the sector.

Curvature-adaptive Perimeter Sampling

The original curve in the object contour between two vertices is approximated by a straight line in the polygon. For regions of high curvature, this is not a good approximation. Thus, we propose an adaptive sampling based on the curvature of the local contour. We distribute the vertices non-uniformly in order to represent the object contour best. Figure 6.4 (right) shows the effectiveness of this approach, where a larger number of vertices is used for higher curvature regions than straight lines, which can be represented by less vertices. We adopt the algorithm in [453] to detect the dominant points in a given curved shape, which best represents the object. Then we reduce the set of points using the algorithm in [454] to get the most representative simplified curves. This way, the polygon has dense points on the curved parts and sparse points on the straight parts, which maximize the utilization of the predefined number of points per contour.

6.3 FisheyeYOLO Network Architecture

We adapt the YOLOv3 [234] model to output different representations as discussed in Section 6.2. We call this model FisheyeYOLO, as illustrated in Figure 6.5. The baseline bounding box model is the same as YOLOv3 [234], except that we replace the Darknet53 encoder with a ResNet18 encoder. Similar to YOLOv3, object detection is performed at multiple scales. For each grid in each scale, object width $(\hat{w})$, height $(\hat{h})$, object center coordinates $(\hat{x}, \hat{y})$ and the object class is inferred. Finally, a non-maximum suppression is used to filter out the low confidence detections. Instead of using L_2 loss for categorical and objectness classification, we used standard categorical cross-entropy and binary entropy losses, respectively. The final loss is a combination

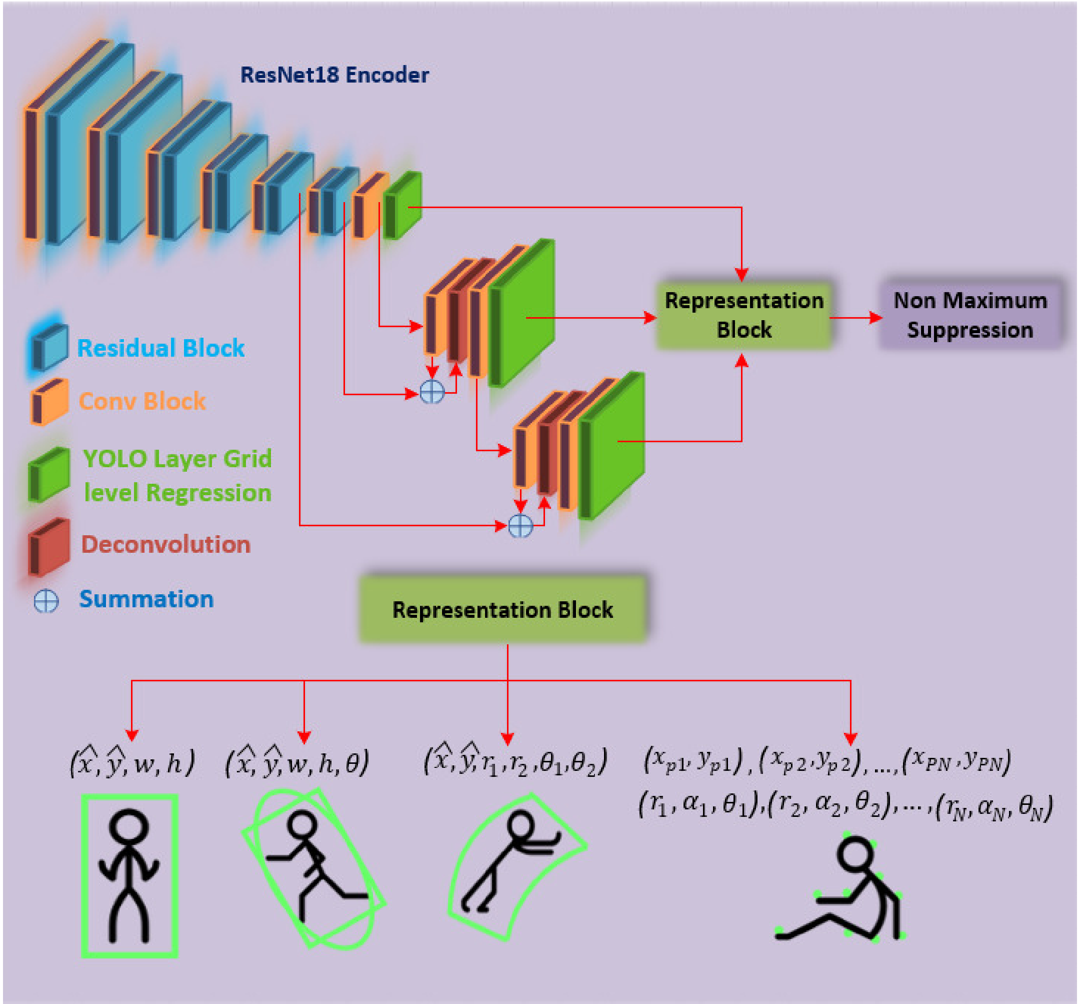

ResNet18 Encoder
Residual Block
Conv Block
YOLO Layer Grid level Regression
Deconvolution
Summation
Representation Block
Non Maximum Suppression
Representation Block
$(\hat{x}, \hat{y}, w, h)$
$(\hat{x}, \hat{y}, w, h, \theta)$
$(\hat{x}, \hat{y}, r_1, r_2, \theta_1, \theta_2)$
$(x_{p1}, y_{p1}), (x_{p2}, y_{p2}), \dots, (x_{PN}, y_{PN})$
$(r_1, \alpha_1, \theta_1), (r_2, \alpha_2, \theta_2), \dots, (r_N, \alpha_N, \theta_N)$

Figure 6.5 **FisheyeYOLO is an extension of YOLOv3** which can output different output representations discussed in Section 6.2.

of sub-losses:

$$\mathcal{L}_{xy} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \tag{6.5}$$

$$\mathcal{L}_{wh} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \tag{6.6}$$

$$\mathcal{L}_{obj} = -\sum_{i=0}^{S^2} \sum_{j=0}^{B} [C_i log(\hat{C}_i)] \tag{6.7}$$

$$\mathcal{L}_{class} = -\sum_{i=0}^{S^2} l_{ij}^{obj} \sum_{c=\text{classes}} [c_{i,j} log(p(\hat{c}_{i,j}))] \tag{6.8}$$

$$\mathcal{L}_{total} = \mathcal{L}_{xy} + \mathcal{L}_{wh} + \mathcal{L}_{obj} + \mathcal{L}_{class} \tag{6.9}$$

where height and width are predicted as offsets from pre-computed anchor boxes.

$$\hat{w} = a_w * e^{f_w} \tag{6.10}$$

$$\hat{h} = a_h * e^{f_h} \tag{6.11}$$

$$\hat{x} = g_x + f_x \tag{6.12}$$

$$\hat{h} = g_y * f_y \tag{6.13}$$

where a_w, a_h anchor box width and height. f_w, f_h, f_x, f_y are the outputs from last layer of the network at grid location g_x, g_y.

In the case of oriented box or ellipse prediction, we define an additional loss function based on ellipse angle or orientation of the box. The loss function for oriented box and ellipse is:

$$\mathcal{L}_{orn} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{obj} [\theta_i - \hat{\theta}_i]^2 \tag{6.14}$$

$$\mathcal{L}_{total} = \mathcal{L}_{xy} + \mathcal{L}_{wh} + \mathcal{L}_{obj} + \mathcal{L}_{class} + \mathcal{L}_{\mathbf{orn}} \tag{6.15}$$

where $\mathcal{L}_{total}$, is the total loss minimized for oriented box regression. In case of curved box, $\mathcal{L}_{wh}$ is replaced by $\mathcal{L}_{cods}$ in Eq. (6.17).

We also explored methods of learning orientation as a classification problem instead of a regression problem. One motivation is due to the discontinuity of angles at 90° due to wrapping around of angles. In this scenario, we discretized the orientation into 18 bins, where each bin represents a range of 10°with a tolerance of +-5°. To further improve the prediction, we design an IoU loss function that guides the model to minimize the difference in the area of the predicted box and the ground truth box. We compute the area of the predicted and ground truth rectangles and apply regression loss on those values. This loss maximizes the overlapping area between the prediction and the ground truth by improving the overall results. The IoU loss is,

$$\mathcal{L}_{IoU} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{obj} [(a_i - \hat{a}_i)^2] \tag{6.16}$$

where a represents the area of the representation at hand. We report all the results related to these experiments in Table 6.4.

The polar polygon regression loss is,

$$\mathcal{L}_{cods} = \sum_{i=0}^{S^2} \sum_{j=0}^{N} \hat{\alpha}_{ij} [(r_{i,j} - \hat{r}_{i,j})^2 + (\theta_{i,j} - \hat{\theta}_{i,j})^2] \tag{6.17}$$

$$\mathcal{L}_{mask} = -\sum_{i=0}^{S^2} \sum_{j=0}^{N} \alpha_{ij} log(\hat{\alpha}_{ij}) \tag{6.18}$$

$$\mathcal{L}_{total} = \mathcal{L}_{xy} + \mathcal{L}_{obj} + \mathcal{L}_{class} + \mathcal{L}_{\mathbf{cods}} + \mathcal{L}_{\mathbf{mask}} \tag{6.19}$$

where N corresponds to the number of sampling points, each point is sampled with a step size of $360/N$ angle in polar coordinates, as shown in Figure 6.4. The polar loss is similar to PolyYOLO [452], where each polygon point is (in red) is represented using

# Vertices	4	12	24	36	60	120
mIoU	85	85.3	86.6	91.8	94.2	98.4

Table 6.1 **Analysis of the number of polygon vertices for representing the objects contour.**

three parameters r, θ, and α. Hence the total required parameters for N sampling points are $3 \times N$. The same is presented in Figure 6.4 (middle).

In Cartesian representation, we regress over two parameters ($\hat{x}$, $\hat{y}$) for each polygon point. We further improve the predictions by adding the IoU loss function, which minimizes the area between the prediction and ground truth. We refer to both loss functions as localization loss $\mathcal{L}_{Localization}$. The combined loss for Cartesian polygon predictions is:

$$\mathcal{L}_{total} = \mathcal{L}_{Class} + \mathcal{L}_{Obj} + \mathcal{L}_{Localization} \tag{6.20}$$

where $\mathcal{L}_{Obj}$ and $\mathcal{L}_{Class}$ are inherited from Yolov3 loss functions. According to the representation at hand, we perform the non-maximum suppression. We generate the predictions for all the representations; filter out the low confidence objects—computation of IoU of the output polygon with the list of outputs where high-IoU objects are filtered out.

6.4 Experimental Results

The objective of this work is to study various representations for the output of fisheye object detection. Conventional object detection algorithms evaluate their predictions against their ground truth, which is usually a bounding box. Unlike conventional evaluation, the first objective is to provide a better representation than a conventional bounding box. Therefore, we first evaluate the representations against the most accurate representation of the object, the ground-truth instance segmentation mask. We report the mIoU between a representation and the ground-truth instance mask.

Additionally, we qualitatively evaluate the representations in obtaining object intersection with the ground (footpoint). This is critical as it helps to localize the object in the map and provides more accurate vehicle trajectory planning. Finally, we report model speed in terms of frames-per-second (fps) as we focus on real-time performance. The distortion is higher in side cameras compared to front and rear cameras. Thus, we provide the evaluation on each camera separately. To simplify the baseline, we only evaluate the vehicle's class, although four classes are available in the dataset.

Number of Polygon Points

The polygon is a more generic representation of complex object shapes that arise in fisheye images. We perform a study to understand the effect of the number of vertices parameter in a polygon. We use a uniform perimeter sampling method to vary the number of vertices and compare the IoU using instance segmentation as ground truth. The results are tabulated in Table 6.1 and the mIoU is calculated between the approximated polygon and the ground truth instance segmentation mask. A 24-sided polygon seems to provide a reasonable trade-off between the number of parameters and model accuracy. Although a 120-sided polygon provides a better ground truth with far too many points, it will be challenging to learn this representation, and it might even produce noisy overfitting. For the quantitative experiments, we fix the

Representation	**mIoU**				**mIoU**	*No. of params*
	Front	Rear	Left	Right		
Standard Box	53.7	47.9	60.6	43.2	51.35	4
Curved Box	53.7	48.6	63.5	44.2	52.5	6
Oriented Box	55.0	50.2	64.8	45.9	53.9	5
Ellipse	56.5	51.7	66.5	47.5	55.5	5
4-sided Polygon (uniform)	70.7	70.6	70.2	69.6	70.2	8
24-sided Polygon (uniform)	85.0	84.9	83.9	83.8	84.4	48
24-sided Polygon (adaptive)	**87.2**	**87**	**86.2**	**86.1**	**86.6**	48

Table 6.2 **Evaluation of the representation capacity of various representations.**

number of vertices to 24 for each object. We observe no significant difference in fps due to increasing the number of vertices where the models run at 56 fps on a standard NVIDIA TitanX GPU. It is due to the utilization of the YoloV3 [234] architecture, which performs the prediction at each grid cell in a parallel manner.

Evaluation of Representation Capacity

Table 6.2 compares the performance of different representations using its ground truth fit relative to the instance segmentation ground truth. We estimate the best fit for each representation using ground truth instance segmentation masks and then compute the mIoU to evaluate capacity. We also list the number of parameters used for each representation to provide a comparison w.r.t. complexity. This empirical metric demonstrates the maximum performance a representation can achieve regardless of the model complexity. As expected, a 24-sided polygon achieves the highest mIoU showing that it has the best representation capacity. The proposed curvature-adaptive polygon achieves a 2.2% improvement over the uniform sampling polygon with the same vertices. Polygon annotation is more expensive to collect, and it increases model complexity. Thus it is still interesting to also consider simple bounding box representations.

Compared to the standard box representation, oriented box representation is approximately 2.5-4% efficient for the side cameras and 1.3-2.3% for front cameras. The ellipse representation improves the efficiency further by an additional 2% for side cameras and 1-2% for front cameras. The curved box achieves a 1.15% improvement over the standard box. However, it is slightly less than for the oriented box representation due to the constraint that two circular sides of the box share the same circle center, which adds some area inside the polygon, decreasing the IoU. In addition, the curvature is not modeled for the horizontal edges of the box. We plan to explore these extensions in future work to obtain a more optimal curved bounding box and leverage circular arcs' convergence at vanishing points.

The curved box's current version has the advantage of getting a tight bottom edge, capturing the footpoint for estimating the object's 3D location. The object's footpoint is captured almost entirely, as observed in qualitative results, especially for the side cameras where distortion is higher. This is important from an application perspective for vehicle navigation, as the footpoint is used for the projection of the object to the 3D world. Compared to polygon representation, curved-box representation has low annotation cost due to fewer representation points, which saves annotation effort.

Representation	**IoU**				**mIoU**
	Front	Rear	Left	Right	
YoloV3	32.5	32.1	34.2	27.8	31.6
Curved Box	33.0	32.7	35.4	28.0	32.3
Oriented Box	33.9	33.5	37.2	30.1	33.6
Ellipse	35.4	35.4	40.4	30.5	35.4
24-sided Polygon	**44.4**	**46.8**	**44.7**	**42.7**	**44.65**

Table 6.3 **Quantitative results of the proposed model using different bounding box representations on the WoodScape dataset.**

Representation	**mAP**
Oriented Box	
Orientation regression	39.0
Orientation classification	40.6
Orientation classification + IoU loss	**41.9**
24-sided Polygon	
Uniform Angular	55.6
Uniform Perimeter	55.4
Adaptive Perimeter	**58.1**

Table 6.4 **Ablation study on the number of parameters in the oriented bounding box and the 24-point polygon representation.**

Quantitative Results

Table 6.4 shows our studies on methods predicting the orientation of the box or the ellipse efficiently. Angle classification and the added IoU loss significantly improved the mAP score relative to a standard baseline. The proposed variable step polygon representation provides a significant improvement of 2.7%. At first, we train a model to regress over the box and its orientation, as specified in Eq. (6.15). In the second experiment, orientation prediction is addressed as a classification problem instead of regression as a possible solution to the discontinuity problem. We divide the orientation range of 180° into 18 bins, where each bin represents 10°, making this an 18 class classification problem. During inference, an acceptable error of +-5 degrees for each box is considered. Using this classification strategy, we improve performance by 1.6%. We are formulating orientation of box or ellipse prediction as a classification model where the IoU loss is found to be superior in performance compared to a direct regression. It yields a 2.9% improvement in accuracy. Hence we use this model as a standard representation for oriented box and ellipse prediction when comparing with other representations.

Table 6.3 demonstrates the prediction results on the proposed representations. The experiments are performed on the best performing model according to Table 6.2 and Table 6.4. Compared to the standard bounding box approach, the proposed oriented box and ellipse models improve the mIoU score on the test set by 2%, 1.8% respectively. Ellipse prediction provides slightly better accuracy than the oriented box as it is not as sensitive to occlusions with other objects in the scene due to the absence of corners, which is demonstrated in Figure 6.7.

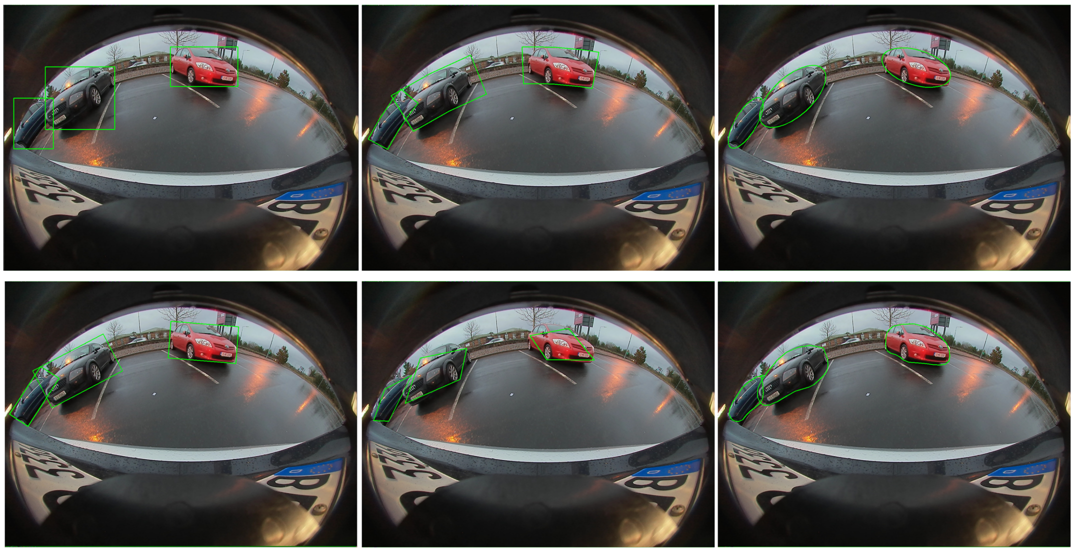


Figure 6.6 **Qualitative results of the proposed model for different output representations.** The 1st row shows the Standard box, Oriented box and Ellipse representations. The 2nd row shows the Curved box, 4-point polygon and 24-point polygon representation.

Figure 6.6 and Figure 6.7 shows a visual evaluation of the proposed representations. Results show that the ellipse provides a decent easy-to-learn representation with a minimum number of parameters and minimum occlusion with the background objects compared to the oriented box representation. Unlike boxes, it allows a minimal representation for the object due to the absence of corners, which for instance, avoids incorrect occlusion with free parking slots, as shown in Figure 6.6 (Bottom). Polygon representation provides higher accuracy in terms of IoU with the instance mask. A four-point model provides high accuracy predictions with small objects as 4 points are sufficient to represent them. As the dataset contains a lot of small objects this representation demonstrates a good accuracy, which is shown in Tables 6.2 and 6.4. Visually, large objects cannot be represented by a quadrilateral, as illustrated in Figure 6.7. A higher number of sampling points on the polygon results in higher performance. However, the predicted masks are still prone to deformation due to minor errors in each point's localization.

6.5 Conclusion

In this chapter, we studied various representations for fisheye object detection. At a high level, we can split them into bounding box extensions and generic polygon representations. We explored the oriented bounding box and ellipse representations. Additionally, we designed a curved bounding box with optimal fisheye distortion properties. We proposed a curvature adaptive sampling method for polygon representations, which improves significantly over uniform sampling methods. Overall, the proposed model improves the relative mIoU accuracy significantly by 40% compared to a YOLOv3 baseline. We consider our method to be a baseline for further research into this area and will make the dataset with ground truth annotation for various representations publicly available. We hope this encourages further research in this area leading to a more mature object detection on raw fisheye imagery.

The final chapter will look into a holistic scene understanding of an autonomous car's environment using monocular fisheye camera videos. All the previous sensory sub-systems' perception algorithms will be integrated into a single fully functional real-time capable system. We will develop a framework that can reason jointly about geometry, motion, and semantics in order to estimate depth accurately, semantic segmentation and motion segmentation and localize in the real world with 2D object detection.

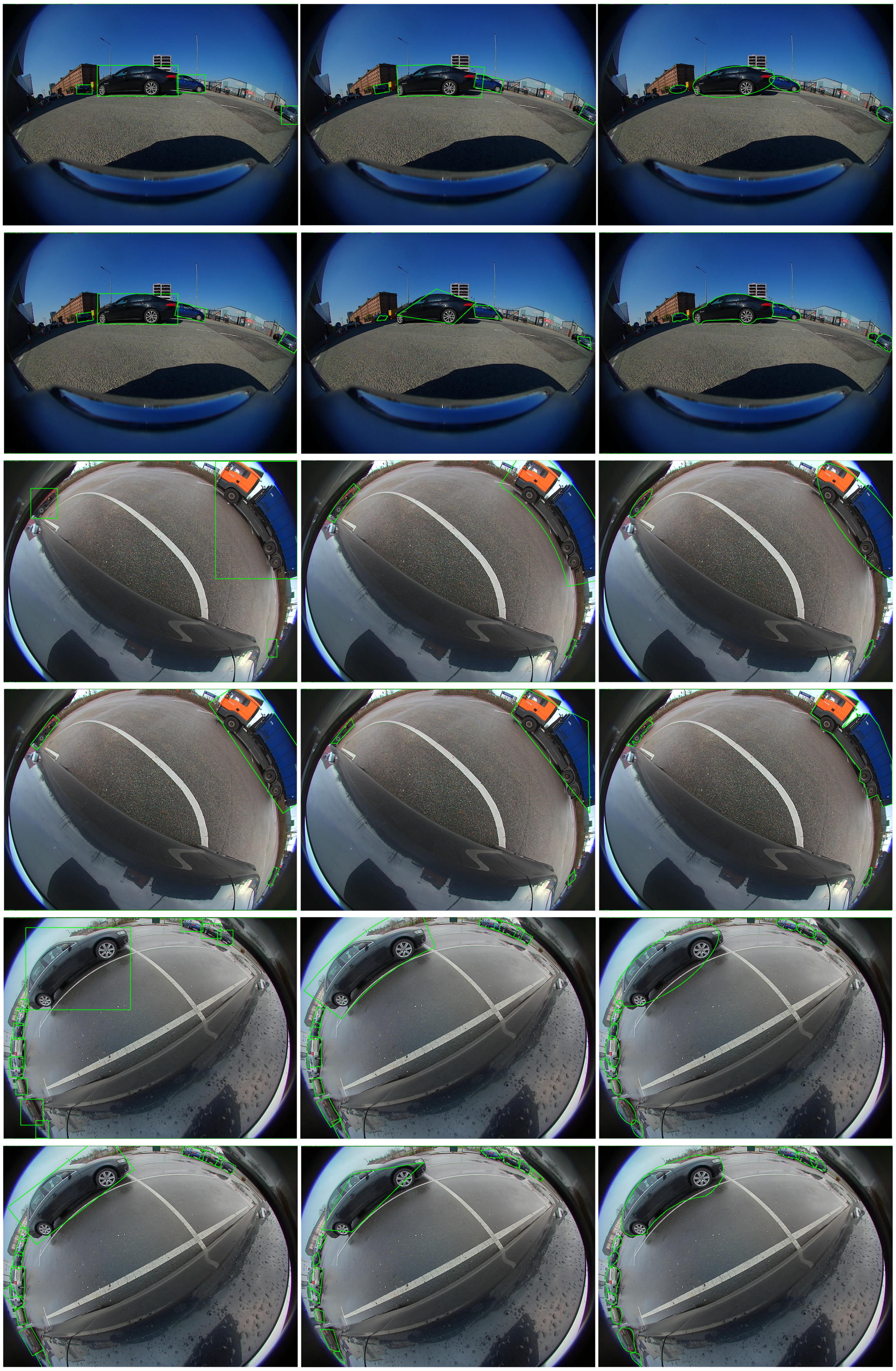


Figure 6.7 **Further qualitative results of the proposed model for different output representations.** The 1st, 3rd and 5th rows show outputs for the Standard box, Oriented box and Ellipse representations. The 2nd, 4th and 6th rows indicate Curved box and 4-point polygon. 24-point polygon representations. For more qualitative results, we refer to this video: `https://youtu.be/iLkOzvJpL-A`.

Chapter 7

Holistic 360° Scene Understanding

Contents

7.1 Introduction

Surround-view fisheye cameras have been deployed in premium cars for over ten years, starting from visualization applications on dashboard display units to providing near-field perception for automated parking. Fisheye cameras have a strong radial distortion that cannot be corrected without disadvantages, including reduced FoV and re-sampling distortion artifacts at the periphery [3]. Appearance variations of objects are larger due to the spatially variant distortion, particularly for close-by objects. It has been paramount to autonomous systems to comprehensively understand the surrounding environment using fisheye cameras. This chapter presents a multi-task visual perception network on unrectified fisheye images to enable the vehicle to sense its surrounding environment. It consists of six primary tasks necessary for an autonomous driving system: depth estimation, visual odometry, semantic segmentation, motion segmentation, object detection, and lens soiling detection.

In recent years DNNs have accomplished impressive success in various applications, including autonomous driving perception tasks. On the other hand, current deep neural networks are easily fooled by adversarial attacks. This vulnerability raises significant concerns, particularly in safety-critical applications. As a result, research into attacking and defending DNNs has gained much coverage. This chapter presents a detailed adversarial attack applied to the diverse multi-task visual perception. In the experiments, we consider both white and black box attacks for targeted and untargeted cases while attacking a task and inspecting the effect on all the others, in addition to inspecting the effect of applying a simple defense method.

This work was very influential from a product perspective to win next-generation projects and be influential in the academic community. This work was formally presented as *OmniDet* [1] in a journal and a conference at the RA-L + ICRA in 2021. The ablation on *Adversarial Attacks* [15] was presented at the ITSC in 2021. To encourage further research in developing multi-task perception algorithms, the code was made public on the Github, which proved to be quite popular in the vision community and significantly helped the discernibility of the approach. Further details about the dataset usage and demo code can be found on the WoodScape website `https://woodscape.valeo.com`.

Autonomous Driving applications require various perception tasks to provide a robust system covering a wide variety of scenarios. Alternate ways to detect objects in parallel are necessary to achieve a high level of accuracy. For example, objects can be detected based on appearance, motion, and depth cues. Despite increasing computation power in automotive embedded systems, efficient design is always needed due to the increasing number of cameras and perception tasks. MTL is an efficient design pattern commonly used where most of the computation is shared across all the tasks [198, 319]. Besides, learning features for multiple tasks can act as a regularizer, improving generalization. Recently, Mao *et al.* [352] illustrated that multi-task learning improves adversarial robustness, which is critical for safety applications. In the automotive multi-task setting, MultiNet [136] was one of the first to demonstrate a three task network on the KITTI, and most further works have primarily worked on a three task setting.

7.2 Perception Tasks and Losses in MTL

This thesis's final goal is to build a multi-task model covering the necessary modules in a *Level 3* autonomous driving system for near-field sensing use cases such as Parking or Traffic Jam assist. This chapter builds upon the previous chapters focused on individual tasks. In general, there is minimal work in the area of fisheye perception. Specifically, there is only a research work on multi-task learning: FisheyeMultiNet [455] which discusses a simple three task network.

The perception system comprises semantic tasks, geometric tasks, and lens soiling detection (shown in Figure 7.1). The standard semantic tasks are object detection

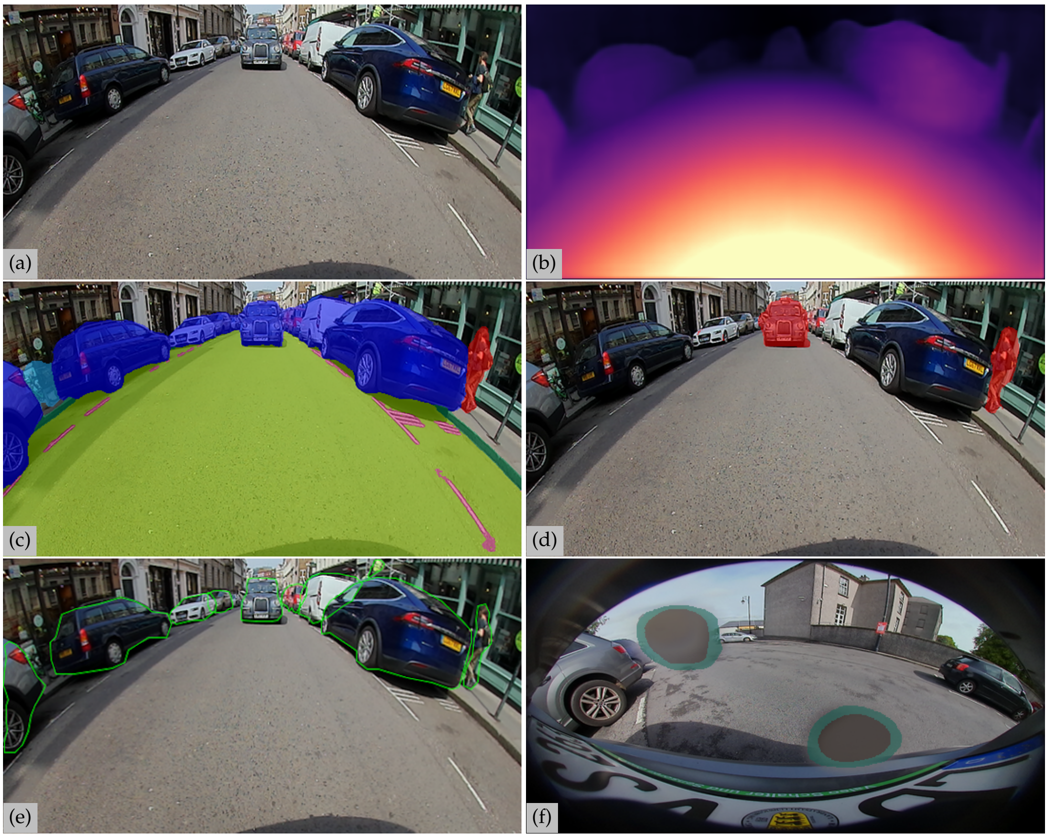


Figure 7.1 **Real-time capable network estimates from the OmniDet framework on raw fisheye images.** (a) Rear-Camera Input Image, (b) Distance Estimate, (c) Semantic Segmentation, (d) Motion Estimation, (e) 24-sided Polygon Object Detection and (f) Soiling Segmentation.

(pedestrians, vehicles, and cyclists) and semantic segmentation (road, lanes, and curbs). Fisheye cameras are mounted low on a vehicle and are susceptible to lens soiling due to splashing of mud or water from the road. Thus, it is vital to detect soiling on the camera lens and trigger a cleaning system [43]. The semantic tasks typically require a large annotated dataset covering various objects. It is practically infeasible practically to cover every possible object. Thus, generic object detection using geometric cues such as motion or depth for rare objects is typically used. They will also complement the detection of standard objects and provide more robustness. Thus we propose to include motion segmentation and depth estimation tasks. Motion is a dominant cue in automotive scenes, and it requires at least two frames or the use of dense optical flow [28]. Self-supervised methods have recently dominated depth estimation, which has also been demonstrated on fisheye images [2]. Finally, the visual odometry task is required to place the detected objects in a temporally consistent map.

7.2.1 Geometric Tasks

Self-Supervised Distance and Pose Estimation Networks

We set up the self-supervised monocular SfM framework following Section 4.3 for distance estimation and pose estimation. View synthesis is performed by incorporating the polynomial projection model from Section 4.3.1. Section 4.3.7 describes the total self-supervised objective loss. Additionally, we include two more loss functions following [181]. To prevent the training objective from getting stuck at multiple local minima for homogeneous areas, we incorporate feature-metric losses computed on I_t's feature representations, where we learn the features using a self-attention autoencoder. The self-supervised loss landscapes are constrained to form proper convergence basins using the first-order $\mathcal{L}_{dis}$ and second-order derivatives $\mathcal{L}_{cvt}$ to regularize the target features. The total objective loss for distance estimation $\mathcal{L}_{dist}$ is calculated by averaging per pixel, scale, and image batch:

$$\begin{aligned}\mathcal{L}_{dist} = \mathcal{L}_r(I_t, \hat{I}_{t' \to t}) + \beta \, \mathcal{L}_s(\hat{D}_t) + \gamma \, \mathcal{L}_{dc}(\hat{D}_t, \hat{D}_{t'}) \\ + \mathcal{L}_r(\hat{F}_t, \hat{F}_{t' \to t}) + \omega \, \mathcal{L}_{dis}(I_t, \hat{F}_t) + \mu \, \mathcal{L}_{cvt}(I_t, \hat{F}_t)\end{aligned} \tag{7.1}$$

where β, γ, ω and μ weigh the distance regularization $\mathcal{L}_s$, cross-sequence distance consistency $\mathcal{L}_{dc}$, discriminative $\mathcal{L}_{dis}$ and convergent $\mathcal{L}_{cvt}$ losses respectively. We calculate the image and feature reconstruction loss using the target I_t, estimated feature $\hat{F}_t$ frames, reconstructed target $\hat{I}_{t' \to t}$ and feature $\hat{F}_{t' \to t}$ frames. It is a linear combination of the general robust pixel-wise loss term [179] and the Structural Similarity (SSIM) [414] as described in [4].

Discriminative loss

Following [181], we define the feature representation by $\varphi_f(uv)$ with gradients $\frac{\partial \varphi(\widehat{uv})}{\partial \widehat{uv}}$ by ensuring that the learned features have relatively large slopes and gradients. For simplicity, the first-order derivative and second-order derivative with respect to image coordinates are denoted by ∇^1 and ∇^2, which equals $\partial_x + \partial_y$ and $\partial_{xx} + 2\partial_{xy} + \partial_{yy}$ respectively. To do so, we constrain the first-order gradients of learned features and formulate it as $-\sum_p |\nabla^1 \varphi(uv)|_1$. As fisheye images have considerably larger homogeneous areas than rectilinear counterparts, this is an essential loss function that penalizes small slopes and emphasizes the low-texture regions using the image gradients. A higher penalty is imposed when similar prominent color areas are encountered.

$$\mathcal{L}_{dis} = -\sum_{uv} e^{-|\nabla^1 I(uv)|_1} |\nabla^1 \varphi(uv)|_1 \tag{7.2}$$

$\mathcal{L}_{dis}$ forces φ to be learned to satisfy the condition that $|\nabla^1 \varphi|_1$ should be relatively large at homogeneous areas, making gradient descent feasible at these regions as these regions receive larger weights. However, merely imposing the discriminative loss cannot guarantee that we move to the optimal solution during the gradient descent.

Merely replacing photometric values with features does not grasp the essence; we wish to have a landscape well-suited for optimization. For deep learning-based methods, whose optimization is mainly based on gradient descent approaches, two main factors of loss functions influence gradient descent's performance. Firstly, the right first-order gradients (slope) to ensure the right optimization direction and the small enough second-order gradients (curvature) to ensure a large convergence radius. However, the commonly used photometric loss cannot fully meet the above two requirements. Due to the raw image intensity limitations, photometric loss tends to have near-zero first-order gradients at low-texture regions. And due to the non-convex property of the image, the convergence radius of the photometric loss is very small (usually 1-2 pixels).

Convergent loss

Since inconsistency exists among first-order gradients, *i.e.*, spatially adjacent gradients point in opposite directions. Shu *et al.* [181] proposed a convergent loss $\mathcal{L}_{cvt}$ to have a relatively large convergence radius to enable gradient descent from a remote distance. This is achieved by formulating the loss to have consistent gradients during the optimization step by encouraging the smoothness of feature gradients and large convergence radii accordingly. Curvature is opposite to convergence radius, *i.e.*; a large curvature corresponds with a small convergence radius, vice versa. This property is of great importance for deep learning based methods; since their learning rates are pre-defined, large learning rates may step over a small convergence radius. $\mathcal{L}_{cvt}$ forces ϕ to have small curvatures, meanwhile encouraging the smoothness of learned feature gradients. $\mathcal{L}_{cvt}$ is formulated to penalize the second-order gradients, *i.e.*,

$$\mathcal{L}_{cvt} = \sum_{uv} |\nabla^2 \varphi(uv)|_1 \tag{7.3}$$

7.2.2 Generalized Object Detection

As discussed in the previous chapter 6, the standard bounding box representation fails in fisheye cameras due to heavy radial distortion, particularly in the periphery. We explored different output representations for fisheye images, including oriented bounding boxes, curved boxes, ellipses, and polygons. We have integrated this model in our MTL framework, where we use a 24-sided polygon representation for object detection. We briefly summarize the details here and refer to the previous chapter 6 for more details on generalized object detection for fisheye camera images. We adapted the YOLOv3 [234] decoder to output polygons as shown in Figure 7.2 and other representations listed above for a uniform comparison.

7.2.3 Segmentation Tasks

Three of the tasks are modeled as segmentation problems. Semantic and soiling segmentation having seven and four output classes, respectively, on the WoodScape dataset. In the following subsections, we briefly look into soiling and motion segmentation tasks.

Soiling Segmentation

As discussed in Section 3.5, in this thesis, we focus on soiling caused by various unwanted particles reposing on the camera lens. These particles' source is mostly mud, dirt, water, or foam created by a detergent. Based on the state of aggregation, such soiling can be either static (*e.g.*, highly viscous mud tends to dry up very quickly, so it does not change its position on the output image over time) or dynamic (mostly water and foam). We want to emphasize that the soiling detection task is necessary for an autonomous driving system as it is used to trigger a camera cleaning system that restores the visibility through the lens [43]. It complements building segmentation or object detection models robust to soiling without an explicit soiling detection step. Coming to the loss function employed, soiling segmentation falls under the supervised learning category and is trained using the *Lovasz-Softmax* [456] loss. The qualitative results of the soiling segmentation are illustrated in Figure 7.3.

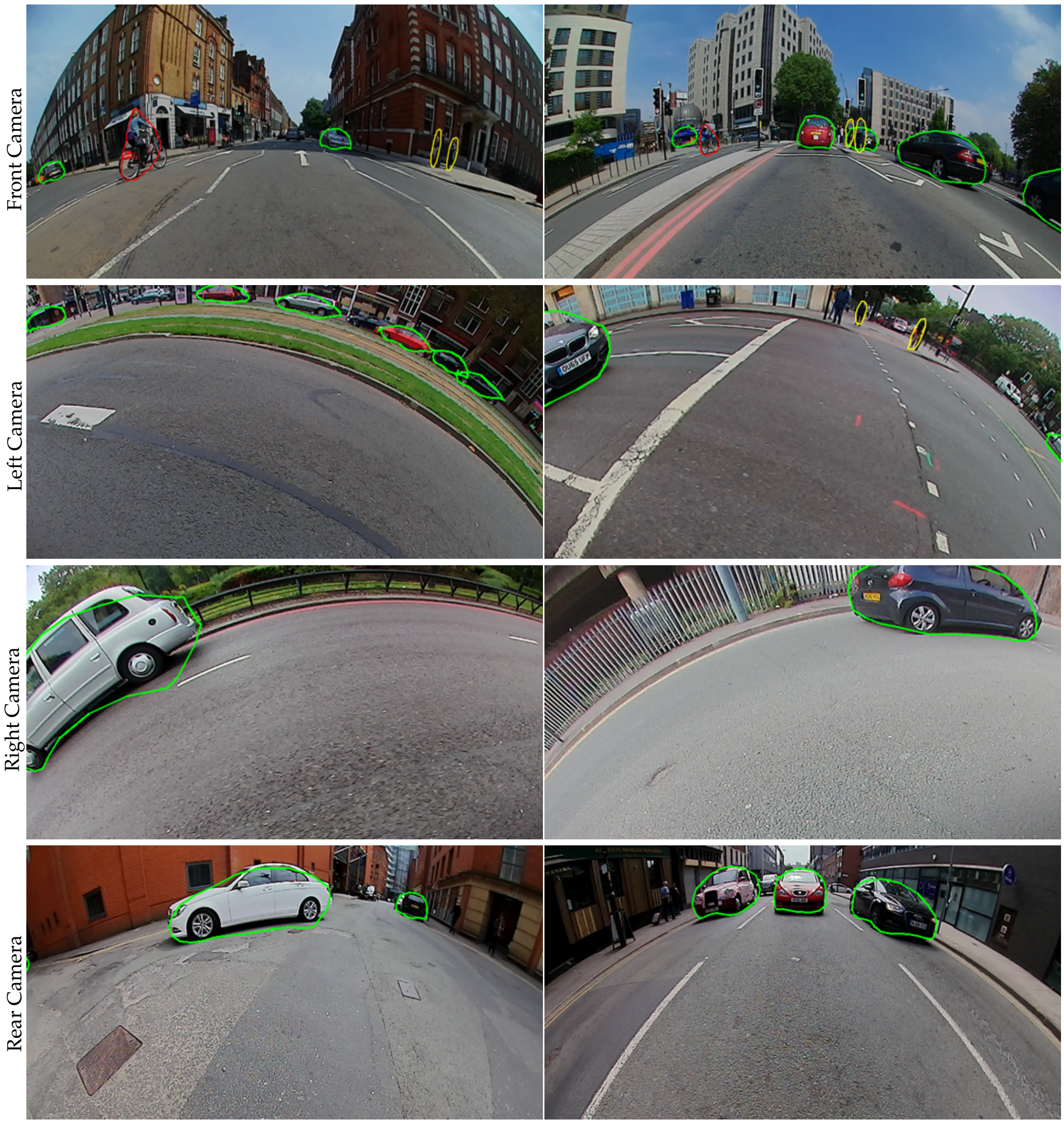


Figure 7.2 **Qualitative results of 24-sided polygon-based objection detection on the WoodScape dataset.**

Motion Segmentation

As discussed in Section 3.4, in this thesis, we propose a CNN architecture for moving object detection using fisheye images that were captured in an autonomous driving environment. Motion segmentation uses two frames and outputs a binary moving or static mask. During training, the network predicts the posterior probability Y_t, which is optimized in a supervised fashion by *Lovasz-Softmax* [456] loss, and *Focal* [216] loss for handling class imbalance instead of the cross-entropy loss. We obtain the final segmentation mask M_t by applying a pixel-wise argmax operation on the posterior probabilities. The qualitative results of the motion segmentation are illustrated in Figure 7.4.

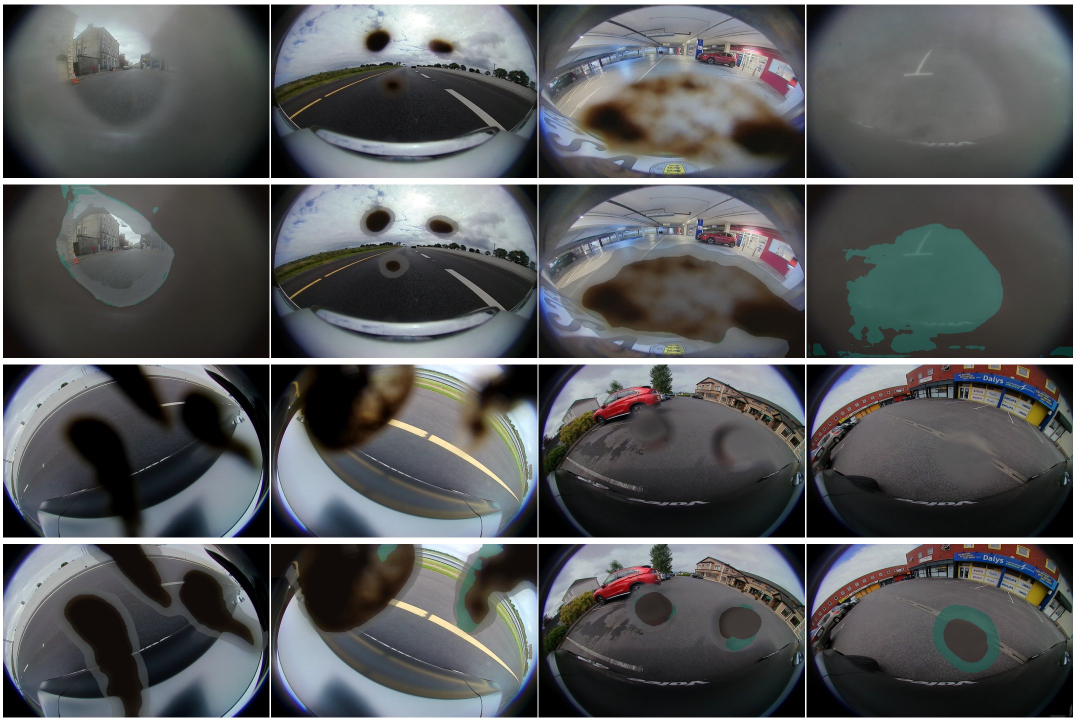
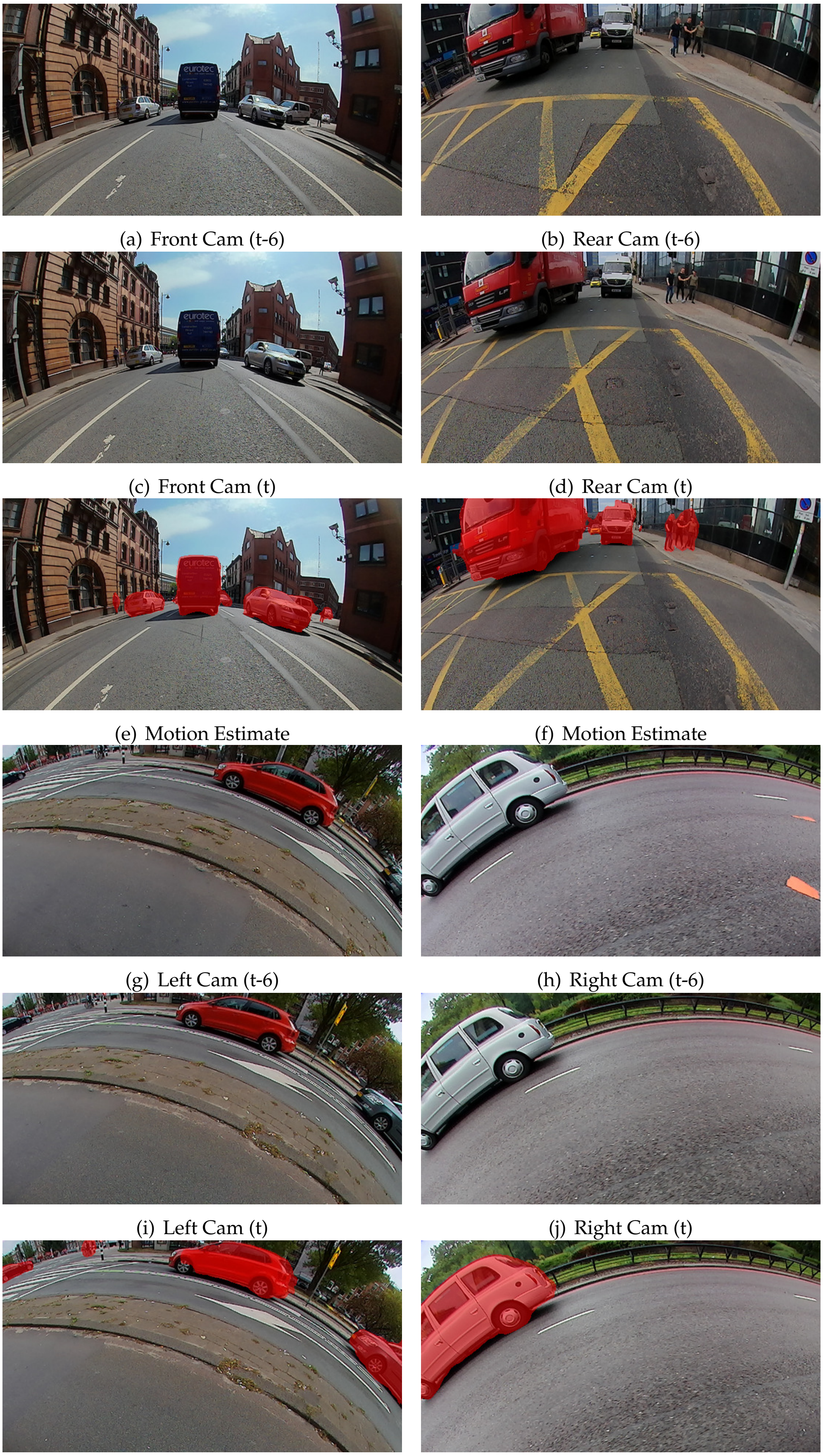


Figure 7.3 **Qualitative results of soiling segmentation on the WoodScape dataset. The 1st and 3rd rows indicate input images from front, rear, left and right cameras. The 2nd and 4th rows indicate the corresponding estimates.**

7.3 Network Details of the OmniDet MTL Framework

Encoder-decoder architectures are commonly used for dense prediction tasks. We use this type of architecture as it easily extends to a shared encoder for multiple tasks. Figure 7.5 provides an overview of the surround-view cameras based multi-task visual perception framework. We design the encoder by incorporating vector attention-based pairwise and patchwise self-attention encoders from [8] as described in Section 5.3.1. These networks efficiently adapt the weights across both spatial dimensions and channels. We adapt the Siamese (twin network) approach for the motion prediction network, where we concatenate the source and target frame features and pass them to the super-resolution motion decoder. As the weights are shared in the Siamese encoder, the previous frame's encoder can be saved and re-used instead of re-computing the features. Inspired by [181], we develop an auxiliary self-attention auto-encoder for single-view reconstruction. We employ the novel CGT described in Section 5.3.2 to handle multiple viewpoints and changes in the camera's intrinsic distance estimation. Secondly, we employ synergized decoders via cross-task connections to improve each other's performance.

Most previous works employed hard parameter sharing techniques, *i.e.*, a shared encoder that branches out into task-specific heads without any synergy. In addition, our goal is to induce synergies across the tasks in the form of loose coupling, maintaining the tasks to be independent. We achieve this by passing some decoder features from one task to another. The decoder features from the guiding task are combined with the intended task's decoder features using pixel-adaptive convolutions that contain an adaptive kernel to mix both feature types. For example, semantic segmentation



(a) Front Cam (t-6)

(b) Rear Cam (t-6)

(c) Front Cam (t)

(d) Rear Cam (t)

(e) Motion Estimate

(f) Motion Estimate

(g) Left Cam (t-6)

(h) Right Cam (t-6)

(i) Left Cam (t)

(j) Right Cam (t)

(k) Motion Estimate

(l) Motion Estimate

Figure 7.4 **Qualitative results of motion segmentation on WoodScape. (t-6) and (t) frames are showcased to visually spot dynamic objects segmented in the motion estimate.**

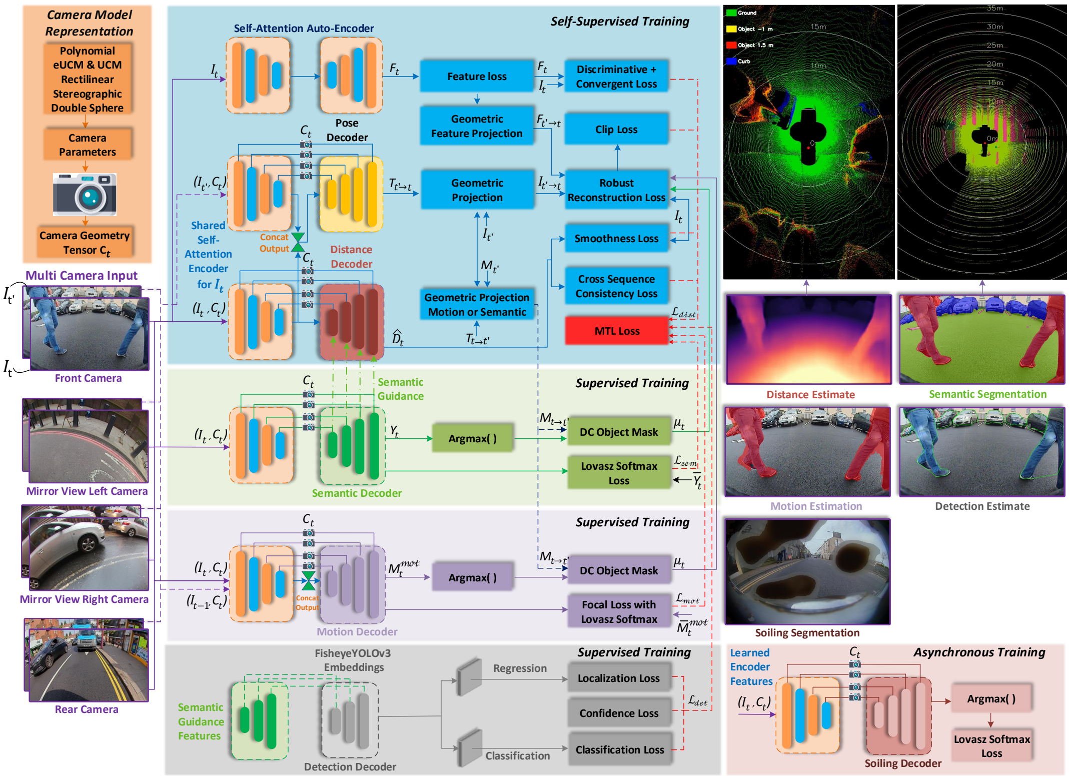


Figure 7.5 **Overview of OmniDet: A surround-view cameras based multi-task visual perception framework.** The distance estimation task (blue block) makes use of semantic guidance and dynamic object masking from semantic/motion estimation (green and blue haze block) and camera-geometry adaptive convolutions (orange block). Additionally, we guide the detection decoder features (gray block) with the semantic features. The encoder block (shown in the same color) is common for all the tasks. The framework consists of processing blocks to train the self-supervised distance estimation (blue blocks) and semantic segmentation (green blocks), motion segmentation (blue haze blocks), polygon-based fisheye object detection (gray blocks), and the asynchronous task of soiling segmentation (rose fog block). We obtain top view geometric information by post-processing the predicted distance and semantic maps in 3D space. The camera geometry tensor C_t (orange block) helps OmniDet to yield distance maps on multiple camera-viewpoints and makes the network camera independent. C_t can also be adapted to the standard camera models, as explained in Section 5.3.2.

produces holes in the road surface due to irregular textures, and depth maps corresponding to the road surface may help regularize the segmentation. There are multiple instances of synergy where semantic segmentation is guiding depth estimation, and object detection features and motion is guiding depth during the training phase. Figure 7.6 illustrates the synergies established in our OmniDet framework.

Dealing With Dynamic Objects and Solving Infinite Depth Issue

This section discusses how we solved one of the concrete challenges using synergy, namely the dynamic object issue for depth estimation, which contaminates the photometric loss and causes infinite depth during inference. As dynamic objects violate the static world assumption, information about their depth/distance is essential in autonomous driving; else, we would encounter the infinite depth issue during the inference stage and not accurately reconstruct the scene and possibly even oversee

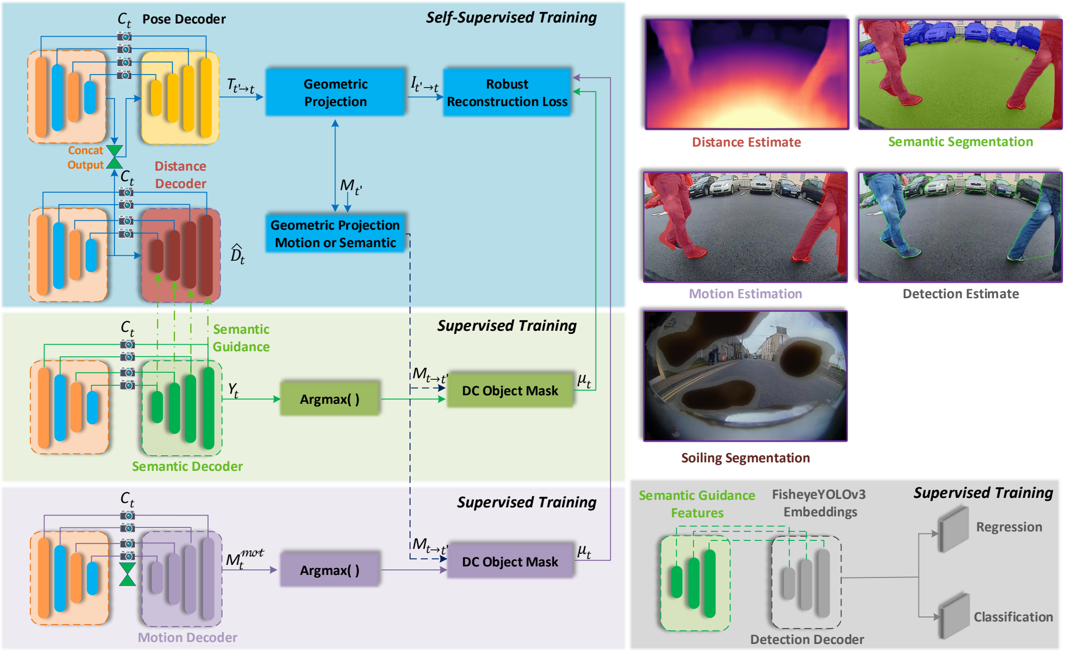


Figure 7.6 **Overview of the synergies established in OmniDet framework.**

other traffic participants. Compared to the previous Section 5.2.3, wherein we use the semantic segmentation task to obtain masks and filter dynamic objects as shown in Figure 7.7. The major drawback of using semantics is that it might not cover all the dynamic object classes in the semantic ground truth (*e.g.*, cows). Therefore, we propose a robust alternative to enable the filtering of dynamic objects using the motion segmentation task as shown in Figure 7.8, which yields either a static or a dynamic mask. However, the user can leverage either one of the tasks to solve this issue. We use the motion segmentation information to exclude potentially *moving* dynamic objects, while the distance is learned from *non-moving* dynamic objects. For this purpose, we define the pixel-wise mask μ_t, which contains a 1 if a pixel does not belong to a dynamic object from the current frame I_t and also not to a wrongfully projected dynamic object from the reconstructed frames $\hat{I}_{t'\to t}$ and a 0 otherwise. Accordingly, we predict a motion segmentation mask M_t^{mot} belonging to the target frame I_t, as well as motion masks $M_{t'}$ for the source frames $I_{t'}$. Dynamic objects inside the source frame are canonically detected inside M_t. However, to obtain the wrongfully projected dynamic objects, we need to warp the motion masks by nearest-neighbor sampling to the target frame, yielding projected motion masks $M_{t'\to t}$. Dynamic objects can be masked through a pixel-wise multiplication of the mask with the reconstruction loss for images and features.

Linking Self-Attention and Semantic features to Distance and Detection decoders

To better incorporate the semantic knowledge extracted from the multi-task networks segmentation branch into the distance estimation, we incorporate it using pixel adaptive convolutions (PAC) described in Section 5.3.3 to distill the knowledge from the semantic features into the distance decoder. This, in particular, breaks up the spatial invariance of the convolutions and allows the incorporation of location-specific semantic knowledge into the multi-level distance features. As shown in Figure 7.5 (green block), the features are extracted at different levels of the segmentation decoder.

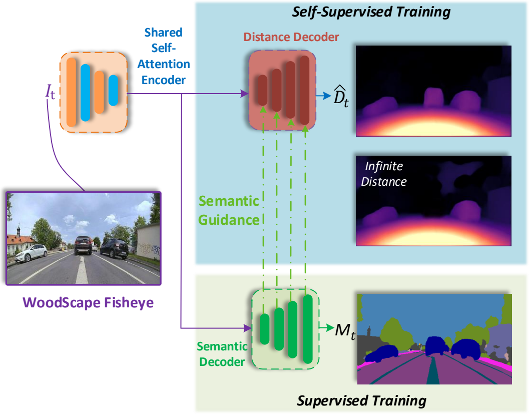


Figure 7.7 **Filtering of dynamic objects using semantic segmentation.**

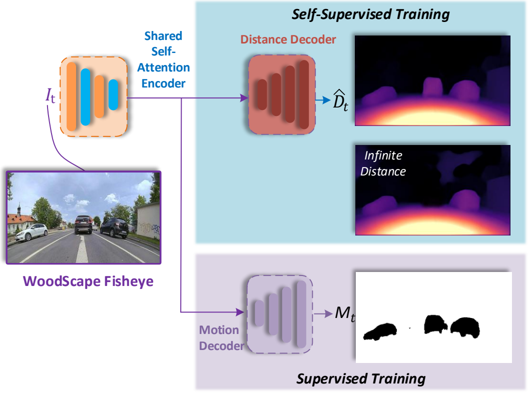


Figure 7.8 **Filtering of dynamic objects using motion segmentation.**

To leverage the multi-task learning setup, at first, we extract the SAN encoder features and feed it as an input signal to the Eq. 5.21 and bypass the spatial information from the SAN encoder to the semantic decoder and fuse these features (skip-connections). Finally, we fuse these features and the detection decoder embeddings by applying PAC and obtaining content-agnostic features. This novel fusion technique in the OmniDet framework significantly improves the detection decoder's accuracy, which can be seen in Table 7.3.

7.3.1 Joint Optimization

We provided a brief overview on MTL optimization strategies in Section 3.6.5. Balancing the task losses is of significant importance in training a multi-task model. We evaluate various task weighting strategies for five tasks compared to the two task experiments in chapter 3 and chapter 4. We evaluate the uncertainty loss from Kendall [132], the gradient magnitude normalization GradNorm [133], the dynamic task prioritization DTP [375], the dynamic weight average DWA [342] and the geometric loss [355].

Secondly, we propose a novel method called **VarNorm** for variance normalization. It consists of normalizing each loss by its variance over the last n epochs. The loss weight of task i at epoch t is formulated as below:

$$w_i(t) = \frac{1}{\sigma_i(t-1)}, \sigma_i(t) = \frac{1}{n-1}\sum_{k=0}^{n-1}(L_i(t-k) - \overline{L_i})^2 \tag{7.4}$$

where $\overline{L_i}$ is the average of task loss i over the last n epochs. We chose $n = 5$. This method is motivated by the simple idea that the task loss values can be seen as a distribution whose dispersion is its variance. Variance normalization re-scales the dispersion between the different task loss distributions based on the previous n epochs. A large dispersion leads to a lower task weight, whereas a small dispersion to a higher one. Its final effect tends to homogenize the learning speed of tasks. As shown in Table 7.1, equal weighting is the worst, and the multi-task network performs better than the single task networks by using any dynamic task weighting method presented above. We employ the proposed VarNorm method for all the further experiments as it achieved the best results.

Task Weighting	*Distance Estimation*		*Semantic Segmentation*		*Motion Segmentation*		*Object Detection*
	Sq. Rel ↓	**Abs Rel** ↓	**mIoU** ↑	**PA** ↑	**mIoU** ↑	**PA** ↑	**mAP** ↑
Single Task	0.060	0.304	72.5	94.8	68.1	94.1	63.5
Equal	0.058	0.302	70.3	92.7	67.3	93.3	64.6
DTP [375]	0.047	0.281	75.8	95.6	75.3	95.3	67.9
DWA [342]	0.054	0.293	75.4	95.2	74.7	95.1	67.5
Geometric [355]	0.061	0.297	74.2	94.1	73.2	94.3	66.7
GradNorm [133]	0.050	0.283	75.9	95.7	74.9	96.0	67.7
Uncertainity [132]	0.044	0.279	76.1	96.2	75.1	95.8	68.0
VarNorm	**0.046**	**0.276**	**76.6**	**96.4**	**75.3**	**96.1**	**68.4**

Table 7.1 **Comparison of task-weighting methods on the WoodScape dataset. PA denotes pixel accuracy.**

7.4 Implementation Details

We systematically train and test all single and multi-task models on the Woodscape and the pinhole camera datasets KITTI and Cityscapes described in the datasets Section 2.5. We use Pytorch [22] and employ a single-stage learning process for the OmniDet framework to facilitate network optimization. We incorporate the recently proposed SAN in the encoder. Zhao *et al*. [8] proposed two convolution variants, namely *pairwise* and *patchwise*. We mainly use patchwise but perform an ablation study using pairwise self-attention convolutions. We employ the Ranger (RAdam [420] + LookAhead [421]) optimizer to minimize the training objective function. We train the model for 20 epochs, with a batch size of 24 on a 24GB Titan RTX with an initial learning rate of 4×10^{-4} for the first 15 epochs, which is reduced to 10^{-5} for the last five epochs. The sigmoid output σ from the distance decoder is converted to distance with $D = m \cdot \sigma + n$, where m and n are chosen to constrain D between 0.1 and 100 units. Finally, we set β, γ, ω and μ to 10^{-3}. Images are resized to 544×288 px from the native 1MP resolution for WoodScape. In the case of Cityscapes, images are resized to 640×384 px for training and validation, and for KITTI, we resize it to input size of 640×192 px for all the tasks. For the motion segmentation, we use the annotations provided by DeepMotion [49] for Cityscapes and KITTI MoSeg [28], where labels are available only for the cars category.

All images from the surround-view cameras with multiple viewpoints are shuffled thoroughly and fed to the distance and pose networks along with their respective intrinsic to create the camera geometry tensor C_t, as shown in Figure 7.5, and described in Section 5.3.2. The soiling dataset is independently built, and thus it cannot be trained jointly in a traditional manner. Thus we freeze the shared encoder trained using five other tasks and train only the decoder for soiling. This demonstrates the addition of new tasks reusing the encoder features. We also trained soiling jointly using asynchronous backpropagation [138], but it achieved the same accuracy as using the frozen encoder. Compared to the previous work SoilingNet [43], we moved from the tiled output to a pixel-level segmentation.

Dist. & Pose Est.	*Sem. Seg.*	*Mot. Seg.*	*Obj. Det.*	*Soil. Seg.*	***RMSE***	***mIoU Seg.***	***mIoU Mot.***	***mAP Det.***	***mIoU Soil.***	*Infer. (fps)*
WoodScape										
✓	✗	✗	✗	✗	1.681	✗	✗	✗	✗	210
✗	✓	✗	✗	✗	✗	72.5	✗	✗	✗	190
✗	✗	✓	✗	✗	✗	✗	68.1	✗	✗	105
✗	✗	✗	✓	✗	✗	✗	✗	63.5	✗	190
✗	✗	✗	✗	✓	✗	✗	✗	✗	80.8	190
✓	✓	✗	✗	✗	1.442	74.8	✗	✗	✗	143
✗	✓	✗	✓	✗	✗	77.1	✗	67.9	✗	143
✓	✓	✓	✗	✗	1.352	75.5	74.8	✗	✗	69
✓	✓	✓	✓	✗	**1.332**	**76.6**	**75.3**	**68.4**	✗	**60**
KITTI										
✓	✗	✗	✗	NA	4.126	✗	✗	✗	NA	160
✗	✓	✗	✗	NA	✗	67.7	✗	✗	NA	148
✗	✗	✓	✗	NA	✗	✗	68.3	✗	NA	78
✗	✗	✗	✓	NA	✗	✗	✗	80.1	NA	182
✓	✓	✗	✗	NA	3.984	72.1	✗	✗	NA	103
✓	✓	✓	✗	NA	3.892	71.9	71.7	✗	NA	47
✓	✓	✓	✓	NA	**3.859**	**72.4**	**72.2**	**82.3**	NA	**43**
CityScapes										
✓	✗	✗	✗	NA	4.906	✗	✗	✗	NA	156
✗	✓	✗	✗	NA	✗	78.7	✗	✗	NA	132
✗	✗	✓	✗	NA	✗	✗	70.4	✗	NA	64
✗	✗	✗	✓	NA	✗	✗	✗	51.7	NA	167
✓	✓	✗	✗	NA	4.741	79.4	✗	✗	NA	91
✓	✓	✓	✗	NA	4.725	79.1	72.0	✗	NA	36
✓	✓	✓	✓	NA	**4.691**	**81.2**	**72.7**	**53.0**	NA	**31**

Table 7.2 **Comparative study of SAN10-Patch MTL model and the equivalent single-task models on three datasets.** The checkmark legends indicate ✓if the task is activated during training, ✗ deactivated, ✗ no evaluation performed, and NA task not being part of the MTL training.

7.5 Experiments

7.5.1 Single-Task vs Multi-Task Learning

In Table 7.2, we perform an extensive ablation of the proposed framework on all considered datasets. The soiling Segmentation task is indicated NA (Not Applicable) as it is not included in the MTL training regime. Quantitative results from the experiments indicate that a multi-task network with six tasks, five diverse tasks performs better than the single task models along with the proposed synergies. The qualitative results on the raw fisheye streams from the surround-view camera system on the perception tasks are shown in Figure 7.10. For KITTI and CityScapes, we employ the novel VarNorm task weighting technique. With this synergy of perception tasks, we obtain state-of-the-art depth and pose estimation results on the KITTI dataset, as shown in Table 7.5 and Table 7.6 respectively. We infer the models using the TensorRT (FP16bit) on NVIDIA's Jetson AGX platform and report processed frames per second for all the tasks.

7.5.2 Ablation Study of the Contributions

For the ablation analysis of the main features shown in Table 7.3, we consider two variants of the self-attention encoder, namely pairwise and patchwise, as described in Section 5.3.1. First, we replace the L_1 loss with a generic parameterized loss function and test it using the self-attention encoder's patchwise variant. We cap the distance

Network	*Robust loss*	*Feature loss*	*Semantic Guide Dist.*	*Semantic Mask*	*Motion Mask*	*Semantic Guide Det.*	*CGT*	*Cyl Rect.*	***RMSE*** ↓	*δ* < 1.25↑	***mIoU Seg***	***mIoU Mot.***	***mAP Det***
OmniDet (SAN10-patch)	✓	✗	✗	✗	✗	✗	✗	✗	2.153	0.875	73.2	71.8	63.3
	✓	✓	✗	✗	✗	✗	✗	✗	1.764	0.897	73.6	72.3	63.5
	✓	✓	✗	✗	✗	✗	✓	✗	1.681	0.902	74.2	73.5	63.8
	✓	✓	✓	✓	✗	✗	✗	✗	1.512	0.905	74.5	73.4	63.6
	✓	✓	✓	✓	✗	✗	✓	✗	1.442	0.908	74.8	74.0	64.1
	✓	✓	✓	✗	✓	✗	✗	✗	1.397	0.915	75.2	74.3	64.0
	✓	✓	✓	✗	✓	✗	✓	✗	1.352	0.916	75.5	74.8	64.3
	✓	✓	✓	✗	✓	✓	✗	✗	1.348	0.915	75.9	74.9	67.8
	✓	✓	✓	✗	✓	✓	✓	✗	**1.332**	**0.918**	**76.6**	**75.3**	**68.4**
	✓	✓	✓	✗	✓	✓	✓	✓	1.210	0.929	78.9	79.2	74.1
OmniDet (SAN10-pair)	✓	✓	✓	✓	✗	✗	✓	✗	1.492	0.904	74.1	73.1	63.3
	✓	✓	✓	✗	✓	✓	✓	✗	1.321	0.911	75.4	74.6	67.6
	✓	✓	✓	✗	✓	✓	✓	✓	1.272	0.919	77.1	77.4	72.6
OmniDet (SAN19-patch)	✓	✗	✗	✗	✗	✗	✗	✗	2.138	0.880	73.9	72.4	64.7
	✓	✓	✗	✗	✗	✗	✗	✗	1.749	0.903	74.3	73.0	64.8
	✓	✓	✗	✗	✗	✗	✓	✗	1.662	0.906	74.6	74.1	65.2
	✓	✓	✓	✓	✗	✗	✗	✗	1.495	0.910	74.9	73.8	64.9
	✓	✓	✓	✓	✗	✗	✓	✗	1.427	0.916	75.4	74.7	65.5
	✓	✓	✓	✗	✓	✗	✗	✗	1.378	0.918	75.7	75.1	65.3
	✓	✓	✓	✗	✓	✗	✓	✗	1.331	0.922	76.2	75.6	65.9
	✓	✓	✓	✗	✓	✓	✗	✗	1.320	0.927	76.8	76.2	69.6
	✓	✓	✓	✗	✓	✓	✓	✗	**1.304**	**0.931**	**77.4**	**77.0**	**71.5**
	✓	✓	✓	✗	✓	✓	✓	✓	1.177	0.938	80.2	80.5	76.3

Table 7.3 **Ablation study on the effect of our contributions** up to the final OmniDet model on the Woodscape.

Semantic Mask	*Motion Mask*	*CGT*	***RMSE*** ↓	***mIoU Seg***	***mIoU Mot.***	***mAP Det***
✓	✗	✗	1.512	74.5	73.4	63.6
✓	✗	✓	1.442	74.8	74.0	64.1
✗	✓	✗	1.397	75.2	74.3	64.0
✗	✓	✓	1.352	75.5	74.8	64.3

Table 7.4 **Ablation study of dynamic object filtering using semantic and motion segmentation masks.**

estimates to 40m. We achieve significant gains in this setting by attributing better-supervised signal provided by using discriminative features $\mathcal{L}_{dis}$ as described in Section 7.2.1 where incorrect distance values are appropriately penalized with more considerable losses along with the combination of $\mathcal{L}_{cvt}$ in Section 7.2.1 wherein a correct optimization direction is provided. These losses help the gradient descent approaches to transit smoothly towards optimal solutions. When adding the CGT to this setting, we observe a significant increase in accuracy since we train multiple cameras with different camera intrinsics and viewing angles. For the OmniDet framework to be operational in the first place, this an important feature. The aforementioned training strategy makes the network camera-independent and generalizes better to images taken from a different camera.

To achieve synergy between geometry and semantic features, we add semantic guidance to the distance decoder. It helps to reason about geometry and content within the same shared features and to disambiguate photometric ambiguities. To establish a robust reconstruction loss free from the dynamic objects' contamination, we introduce semantic and motion masks as described in Section 7.3, to filter all the dynamic objects. Motion mask-based filtering yields superior gains along with CGT compared to using semantic masks, which is also shown in Table 7.4 as semantics might not contain all the dynamic objects in its set of classes as indicated in Eq. 5.7.

	Method	Abs_{rel}	Sq_{rel}	$RMSE$	$RMSE_{log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
		lower is better				higher is better		
Original [50]	Monodepth2 [55]	0.115	0.903	4.863	0.193	0.877	0.959	0.981
	PackNet-SfM [7]	0.111	0.829	4.788	0.199	0.864	0.954	0.980
	FisheyeDistanceNet [2]	0.117	0.867	4.739	0.190	0.869	0.960	0.982
	UnRectDepthNet [3]	0.107	0.721	4.564	0.178	0.894	0.971	**0.986**
	SynDistNet [4]	0.109	0.718	4.516	0.180	0.896	0.973	**0.986**
	Shu *et al.* [181]	0.104	0.729	4.481	0.179	0.893	0.965	0.984
	OmniDet	**0.092**	**0.657**	**3.984**	**0.168**	**0.914**	**0.975**	**0.986**
	Struct2Depth* [189]	0.109	0.825	4.750	0.187	0.874	0.958	0.983
	GLNet* [364]	0.099	0.796	4.743	0.186	0.884	0.955	0.979
	Shu* *et al.* [181]	0.088	0.712	4.137	0.169	0.915	0.965	0.982
	OmniDet*	**0.077**	**0.641**	**3.859**	**0.152**	**0.931**	**0.979**	**0.989**
Improved [143]	Monodepth2 [55]	0.090	0.545	3.942	0.137	0.914	0.983	0.995
	PackNet-SfM [7]	0.078	0.420	3.485	0.121	0.931	0.986	0.996
	UnRectDepthNet [3]	0.081	0.414	3.412	0.117	0.926	0.987	0.996
	SynDistNet [4]	0.076	0.412	3.406	0.115	0.931	0.988	0.996
	OmniDet	**0.067**	**0.306**	**3.098**	**0.101**	**0.944**	**0.991**	**0.997**
	OmniDet*	**0.048**	**0.287**	**2.913**	**0.081**	**0.948**	**0.991**	**0.998**

Table 7.5 **Evaluation of depth estimation** on the KITTI Eigen [50] split.

Figure 7.9. presents the qualitative results of the distance estimation and semantic segmentation tasks. Finally, to complete the synergy, we use semantically guided features to the detection decoder described in Section 5.3.3, which yields significant gains in mAP, and overall results for all the tasks are inherently improved with better-shared features. All the contributed features and the synergy between tasks help the OmniDet framework to achieve a good scene understanding with high accuracy in each task's predictions. To enable a single model to handle the different intrinsics, we re-projected all input images to the same central cylindrical projection in the first step. In vertical direction, it resembles a rectilinear projection $y_I = f \cdot \tan(\theta')$, where $\theta' = \arctan\left(y/\sqrt{x^2+z^2}\right)$. In horizontal direction it resembles a equirectangular projection $x_I = f \cdot \theta''$, where $\theta'' = \arctan(x/z)$. Cylindrical rectification (Cyl Rect.) provides a good trade-off between loss of field-of-view and reducing distortion [12]. The qualitative results of the Cylindrical rectified image predictions on the perception tasks are shown in Figure 7.11.

7.5.3 State-of-the-Art Comparison on KITTI

To facilitate comparison to previous methods, we also train the distance estimation method in the classical depth estimation setting on the KITTI Eigen split [116] whose results are shown in Table 7.5. With the synergy between depth, semantic, motion, and detection tasks along with the features ablated in Table 7.3 and their importance explained in Section 7.5.2, *we outperform all previous monocular methods.* Following best practices, we cap depths at 80 m. We also evaluate using the *Original* [50] as well as *Improved* [143] ground truth depth maps. Method* indicates the online refinement technique [189], where the model is also trained during inference. Using the online refinement method from [189], we obtain a significant improvement.

In Table 7.6, we report the average trajectory error of the pose estimation network in meters by following the same protocols as Zhou [53] on the official KITTI odometry

Method	*No. of Frames*	*GT*	*Sequence 09*	*Sequence 10*
ORB-SLAM [108]	5	✓	0.014 ± 0.008	0.012 ± 0.011
DF-Net [416]	5	✓	0.017 ± 0.007	0.015 ± 0.009
SfMLearner [53]	5	✓	0.016 ± 0.009	0.013 ± 0.009
Klodt et al. [436]	5	✓	0.014 ± 0.007	0.013 ± 0.009
GeoNet [370]	5	✓	0.012 ± 0.007	0.012 ± 0.009
Struct2Depth [189]	5	✓	0.011 ± 0.006	0.011 ± 0.010
Ranjan [303]	5	✓	0.011 ± 0.006	0.011 ± 0.010
PackNet-SfM [7]	5	✓	0.010 ± 0.005	0.009 ± 0.008
PackNet-SfM [7]	5	✗	0.014 ± 0.007	0.012 ± 0.008
OmniDet	5	✓	**0.009** ± **0.004**	0.008 ± **0.005**
OmniDet	5	✗	**0.010** ± **0.005**	**0.010** ± **0.008**
DDVO [177]	3	✓	0.045 ± 0.108	0.033 ± 0.074
Vid2Depth [154]	3	✓	0.013 ± 0.010	0.012 ± 0.011
EPC++ [363]	3	✓	0.013 ± 0.007	0.012 ± 0.008
OmniDet	3	✓	**0.011** ± **0.006**	**0.010** ± **0.007**
OmniDet	3	✗	**0.012** ± **0.007**	**0.011** ± **0.008**
Monodepth2 [55]	2	✓	0.017 ± 0.008	0.015 ± 0.010
OmniDet	2	✓	**0.015** ± **0.007**	**0.013** ± **0.007**
OmniDet	2	✗	**0.016** ± **0.008**	**0.014** ± **0.009**

Table 7.6 **Evaluation of the pose estimation** on the KITTI Odometry Benchmark [116].

Representation	*mIoU GT*	*mIOU*	*No. of params*
Standard Box	51.3	31.6	4
Curved Box	52.5	32.3	6
Oriented Box	53.9	33.6	5
Ellipse	55.5	35.4	5
24-sided Polygon	**86.6**	**44.6**	**48**

Table 7.7 **Evaluation of various object detection representations.**

split (containing 11 sequences with ground-truth (GT) odometry acquired with the IMU/GPS measurements, which is used for evaluation purpose only), and use sequences 00-08 for training and 09-10 for testing. We outperform the previous methods listed in Table 7.6, mainly by applying the bundle adjustment framework inflicted by the cross-sequence distance consistency loss [2] which induces more constraints and simultaneously optimizes distances and camera poses for an implicitly extended training input sequence. This provides additional consistency constraints that are not induced by previous methods.

For object detection on native fisheye images, in addition to the standard 2D box representation, we benchmark oriented boxes, ellipse, curved boxes, and 24-sided polygon representations in Table 7.7. Here mIoU GT represents the maximum performance we can achieve in terms of instance segmentation by using each representation. It is computed between the ground truth instance segmentation and the ground truth of the corresponding representation. Whereas mIoU represents the evaluation of the performance achieved on the network estimates. We also list the number of parameters involved in the model for each representation to provide a complexity comparison.

7.5.4 Analysis on Adversarial Attacks

We conduct the experiments across four visual perception tasks, excluding the pose estimation and soiling segmentation tasks, on a test set of 100 images, *i.e.*, randomly sampled from the original test set of the target network. We generate adversarial examples for each image in the test set while attacking one task at a time. For white-box attacks, given the available gradients, we perform an iterative optimization process to add perturbation in the input image in a direction to harm the original predictions. For the black box attacks, we set up similar protocols as established for white box attacks; however, the gradients are not given but estimated. As a generic black-box optimization algorithm, we show that Evolution Strategies (ES) can be adopted as a black-box optimization method for generating adversarial examples. Precisely, the ES algorithm is used to update the adversarial example over the attacking steps. At each step, we take the adversarial example vector, *i.e.*, the adversarial example is the perturbed image, and generate a population of 25 slightly different vectors by adding noise sampled from a normal distribution. Then, we evaluate each of the individuals by feeding it through the network. Finally, the newly updated vector is the weighted sum of the population vectors. Each weight is proportional to the task's desired performance, and the process continues till convergence defined by a stopping criterion.

In the untargeted case, the aim is to inflict maximum harm to the predictions without considering a certain target prediction: $f(x_{adv}) \neq y_{true}$, however in the targeted case, the aim is to harm the predictions in a desired specific way towards a certain target: $f(x_{adv}) = y_{target}$. The attack loss is based on the task. Mean square error (MSE) is used for the distance task, while cross-entropy loss is used for motion and semantic segmentation tasks. For the object detection task, only object confidence is attacked; hence the cross-entropy loss is adopted. Regarding untargeted attacks across all the tasks, the goal is to maximize the distance between the original output of the network and the adversarial example's output. Accordingly, we add the perturbations to achieve this simple goal where the output can be anything but the correct one. This can be formulated as $\theta = \theta + \alpha dJ/d\theta$ where θ is the image parameters *i.e.* pixels, J the loss functions and α is the learning rate. However, for the targeted attacks, the target output is defined. The aim is to minimize the distance between the original output and the target output according to $\theta = \theta - \alpha dJ/d\theta$.

For each perception task, the targets are as follows: The *Targeted Depth* attack tries to convert the predicted near pixels to be predicted as far. The *Targeted Segmentation* attack tries to convert the predicted vehicle pixels as void for randomly 50% of the test set. For the other 50%, the attack tries to convert the predicted road pixels to void. The *Targeted Motion* attack tries to convert the predicted dynamic object pixels as static. Finally, similar to semantic segmentation, the *Targeted Object Detection* attack tries to increase or decrease the predicted confidence randomly. In addition to attacks, we apply a simple blurring defense approach across all attacks. Similar to [403], the intuition is to try to remove the adversarial perturbations and restore the original output as much as possible. The hyperparameters of the attacks are empirically defined based on a minimal validation set of three samples. All white-box attacks are conducted with a learning rate of $\alpha = 0.00015$. In black-box attacks, hyperparameters are chosen to balance the attack effect and the severity of the perturbations. The learning rates range from 0.0001 to 0.001, and $\mu = 0, \sigma = 0.05$ for ES population generation.

Adversarial attacks results

In this subsection, we present and discuss the results of w.r.t. adversarial attacks. As expected, white-box attacks, where the gradients are accessible, were easier to optimize than the black box case. White box attacks can generate adversarial examples with minimal and localized perturbations across all tasks. On the other hand, ES black-box attacks have more significant perturbations and require more hyperparameters to optimize.

The attacking curves for white and black box attacks are shown in Figures 7.12 and 7.13 respectively. Each plot shows each perception task's performance over the 50 attacking steps where the first step at index 0 represents the actual performance of the target network without applying any attack. Each curve shows the mean performance of a task over the test set, where the shaded area is the mean ± standard deviation. Generally, motion and detection tasks have a performance with a large standard deviation indicating the test set's diversity containing easy and hard examples. Across all curves, it is clear that the performance is decreasing along with the attacking steps.

Moreover, attacking one task by generating an adversarial example affects the other tasks' performance in different ways along the attacking curve. These curves enable the adversary to decide at which step the adversarial example is generated according to the required effect on the target task and the other tasks. As shown in Figures 7.12 and 7.13, in most cases, attacking other tasks has a marginal negative effect on the motion task. The main reason is that the motion task takes two frames as input, and only one of them is attacked. Moreover, it is shown that attacking the distance task affects both segmentation and detection tasks. Attacking segmentation or detection showed to affect all other tasks. As mentioned, the attack effect depends on the parameters selected for the attack. Moreover, targeted attacks try to optimize the adversarial example to produce the required target prediction. In contrast, the untargeted attack continues to apply perturbations to produce as different as possible predictions.

To understand the effect of applying a defense method on the attacks, Gaussian blurring with a radius = 1 is applied to the final adversarial examples, which are then fed into the target network. As shown in Table 7.8, this simple defense method has a positive effect for both segmentation and motion tasks in most cases compared to depth and detection tasks. Furthermore, the effect of blurring on the network's performance is inspected without applying any attacks, as shown in Table 7.9. Both detection and distance tasks are affected the most. This explains why this defense method is more effective for segmentation and motion tasks. Figure 7.14 shows different visual samples of the attacks organized into four groups. Each group has three images: the original output, the adversarial perturbations magnified to 10X, and the impacted results are overplayed on the adversarial examples. As expected, perturbations for the white box attacks are much more minor and more localized than the black box case. Moreover, the performance is harmed for the untargeted attacks without having a specific goal leading to arbitrary predictions. On the other hand, vehicles or roads are removed for the semantic segmentation task for targeted attacks. We add false objects or remove true objects for the detection task. Near pixels are converted as far for the distance task. Finally, we convert dynamic objects to static for the motion task.

Task		*Distance* *RMSE*		*Segmentation* *mIoU*		*Motion* *mIoU*		*Detection* *mAP*	
		A	D	A(%)	D(%)	A(%)	D(%)	A(%)	D(%)
Distance	*wb_untarget*	0.126	0.047	-14	-7.3	-3.1	-3.0	-13.4	-25.5
	wb_target	0.288	0.031	-40	-7.5	-4.4	-2.7	-38.9	-33.2
	bb_untarget	0.036	0.033	-3.0	-6.5	-1.5	-3.7	-4.6	-25.8
	bb_target	0.035	0.036	-14.8	-13.4	-3.9	-2.9	-27.1	-37.6
Segmentation	*wb_untarget*	0.032	0.028	-86.8	-14.2	-5.0	-3.5	-37.6	-30.1
	wb_target	0.017	0.027	-32.0	-5.5	-4.0	-2.6	-21.3	-27.5
	bb_untarget	0.015	0.031	-26.1	-11.4	-2.3	-4.9	-6.7	-27.3
	bb_target	0.020	0.034	-16.1	-9.2	-2.2	-2.5	9.2	-28.8
Motion	*wb_untarget*	0.018	0.027	-11.1	-7.3	-25.9	-9.2	-18.9	-23.1
	wb_target	0.010	0.027	-2.4	-6.0	-14.7	-9.2	-7.1	-30.0
	bb_untarget	0.030	0.039	-17.1	-15.8	-24.3	-17.6	-22.2	-38.6
	bb_target	0.033	0.040	-24.6	-22.5	-13.9	-11.7	-34.7	-47.9
Detection	*wb_untarget*	0.012	0.027	-5.1	-5.9	-2.1	-2.5	-39.8	-31.4
	wb_target	0.018	0.027	-15.0	-6.2	-3.0	-4.0	-71.9	-30.6
	bb_untarget	0.021	0.033	-12.5	-10.4	-4.2	-5.8	-39.4	-35.3
	bb_target	0.022	0.034	-11.6	-10.6	-2.7	-3.7	-34.4	-37.9

Table 7.8 **Summary of attacking and defending results** across the test data where *A* and *D* columns are for Attack and Defense respectively.

Task	*Metric*	*Original*	*Blurred*	*Effect (%)*
Distance	RMSE	0.0	0.026	NA
Segmentation	mIoU	0.499	0.477	-4.4
Motion	mIoU	0.711	0.693	-2.6
Detection	mAP	0.633	0.416	-34.3

Table 7.9 **Input blurring effect on the tasks.**

7.6 Conclusion

This chapter successfully demonstrated a six-task network with a shared encoder and synergized decoders on fisheye surround-view images in this thesis. The majority of the automated driving community's research continues to focus on individual tasks, and there is much progress to be made in designing and training optimal multi-task models. We introduced several novel contributions, including the camera geometry tensor usage for encoding radial distortion, variance-based normalization task weighting, and generalized object detection representations. To enable a comparison, we evaluate the network on five tasks on KITTI and Cityscapes, achieving competitive results. There are still many practical challenges in scaling to a higher number of tasks: building a diverse and balanced dataset, corner case mining, stable training mechanisms, and designing an optimal map representation that combines all tasks, thereby reducing the post-processing. We hope that this thesis encourages further research in building a unified perception model for autonomous driving.

This chapter also applied various adversarial attacks to understand the vulnerabilities of the perception network. For each perception task, white and black box attacks are conducted for targeted and untargeted scenarios. Moreover, the attacking curves show the interactions between attacks on different tasks. It is shown how attacking a task affects not only that task but also the others. Moreover, by applying blurring on the adversarial examples as a defense method, it is found to positively affect segmentation and motion tasks in contrast to the object detection and distance estimation tasks for the considered network model.

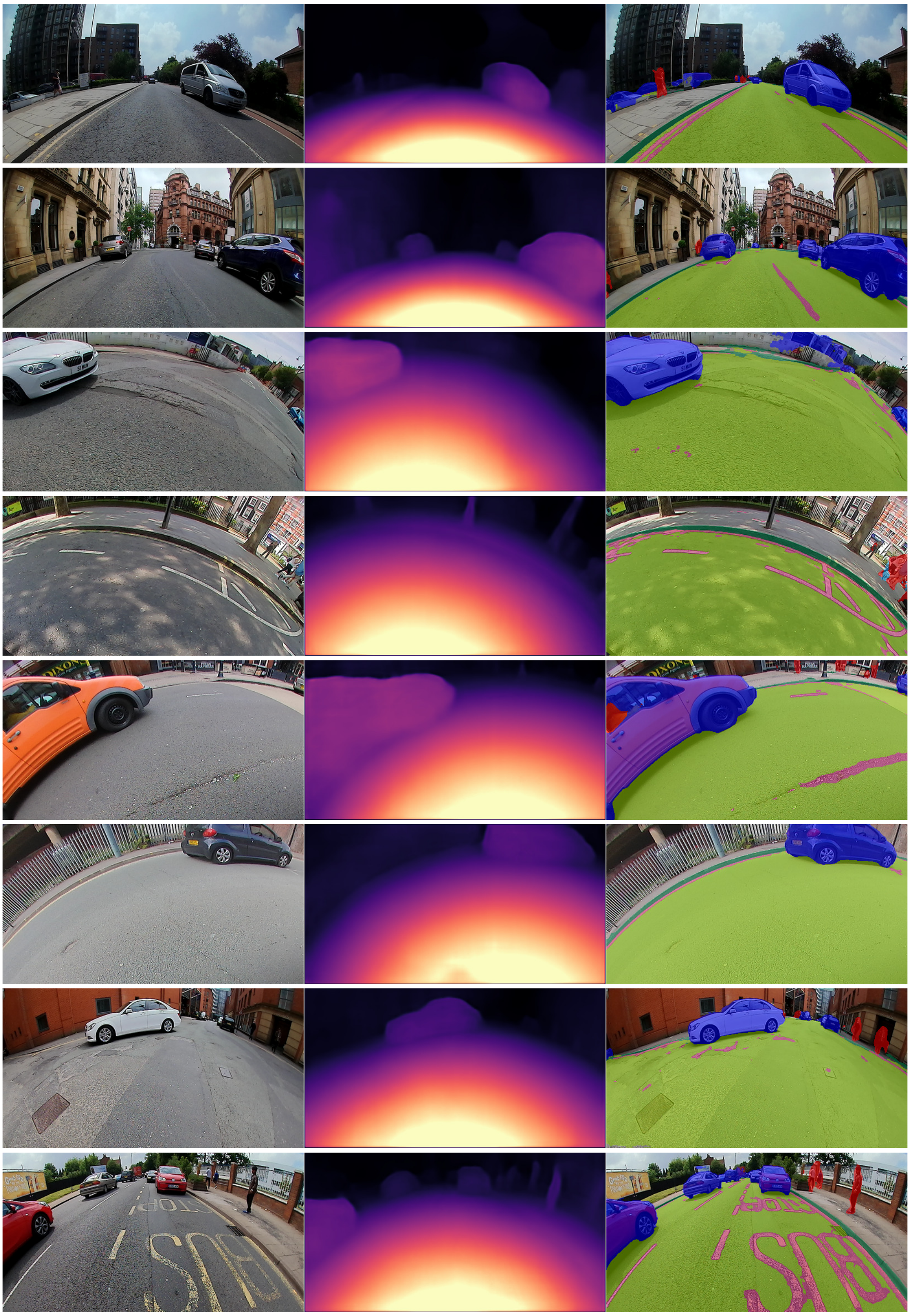


Figure 7.9 **Qualitative results of distance estimation and semantic segmentation on raw surround-view fisheye cameras on the WoodScape dataset. The 1st column contains two rows of input images from each of the Front, Left, Right, and Rear cameras.**

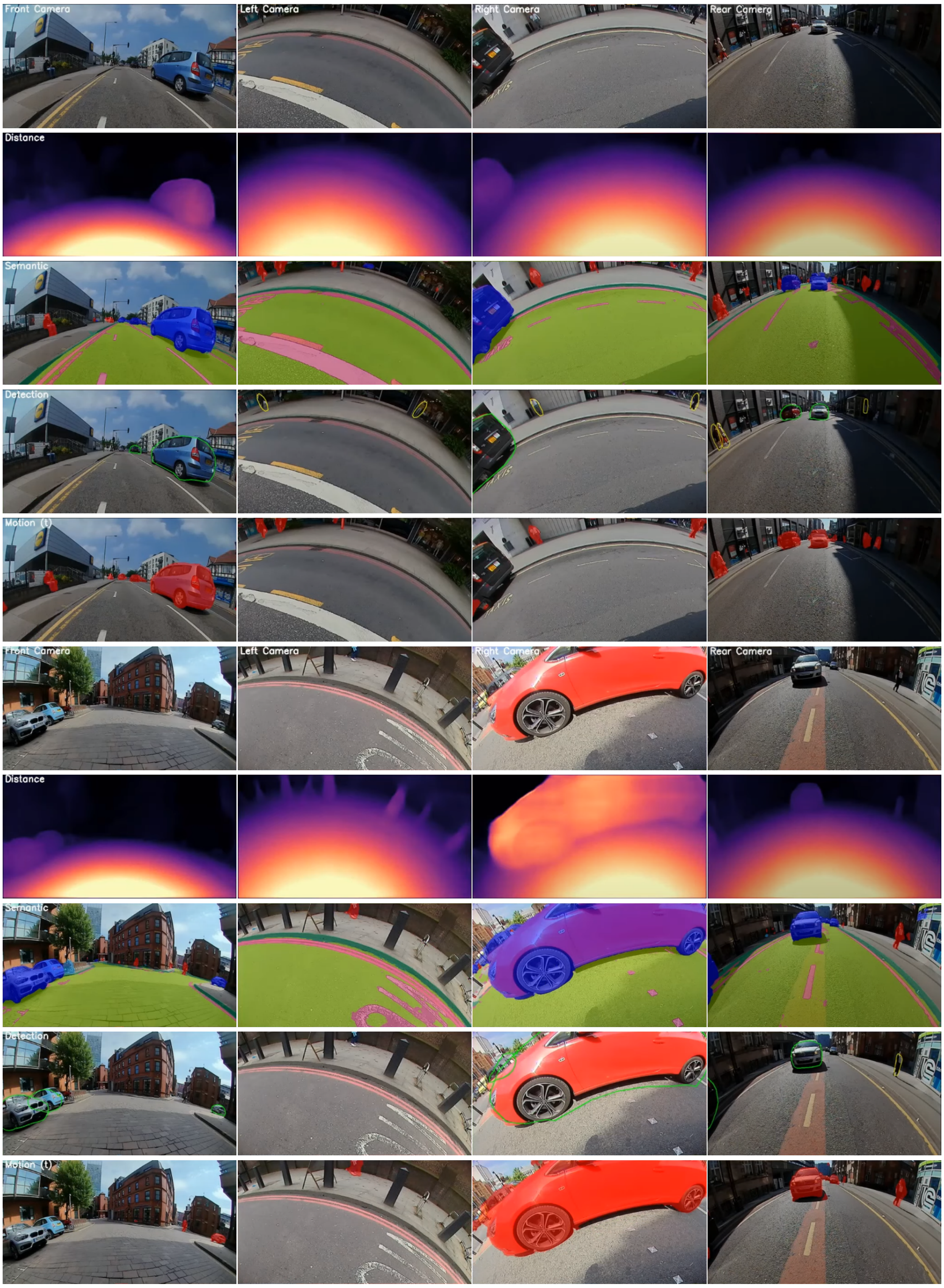


Figure 7.10 **Qualitative results of raw fisheye images from the OmniDet framework on the WoodScape dataset.** The 1st and 6th rows indicates the input images from Front, Left, Right and Rear cameras. 2nd and 7th rows indicate distance estimates, 3rd and 8th rows indicate semantic segmentation maps, 4th and 9th rows indicate generalized object detection predictions and finally 5th and 10th rows indicate the motion segmentation. For more qualitative results at a higher resolution, we refer to this video: https://youtu.be/xbSjZ5OfPes

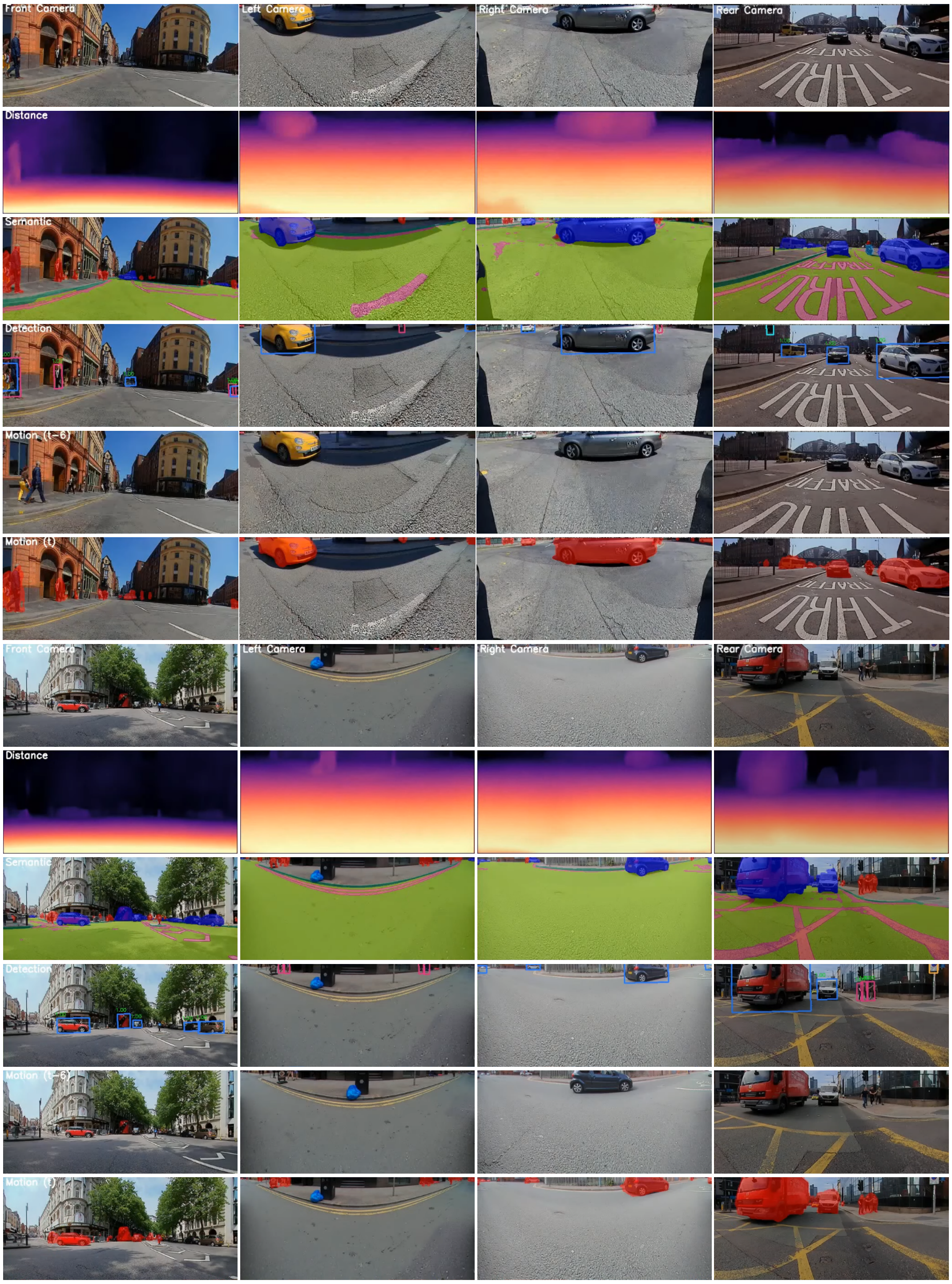


Figure 7.11 **Qualitative results of cylindrical rectified images from the OmniDet framework on the WoodScape dataset.** The 1st and 7th rows indicates the input images from Front, Left, Right and Rear cameras. 2nd and 8th rows indicate distance estimates, 3rd and 9th rows indicate semantic segmentation maps, 4th and 10th rows indicate object detection of standard boxes representations, and finally 5th and 11th rows show input frames at (t-6) for reference to compare the motion segmentation in 6th and 12th rows at time (t). For more qualitative results at a higher resolution, we refer to this video: https://youtu.be/xbSjZ5OfPes

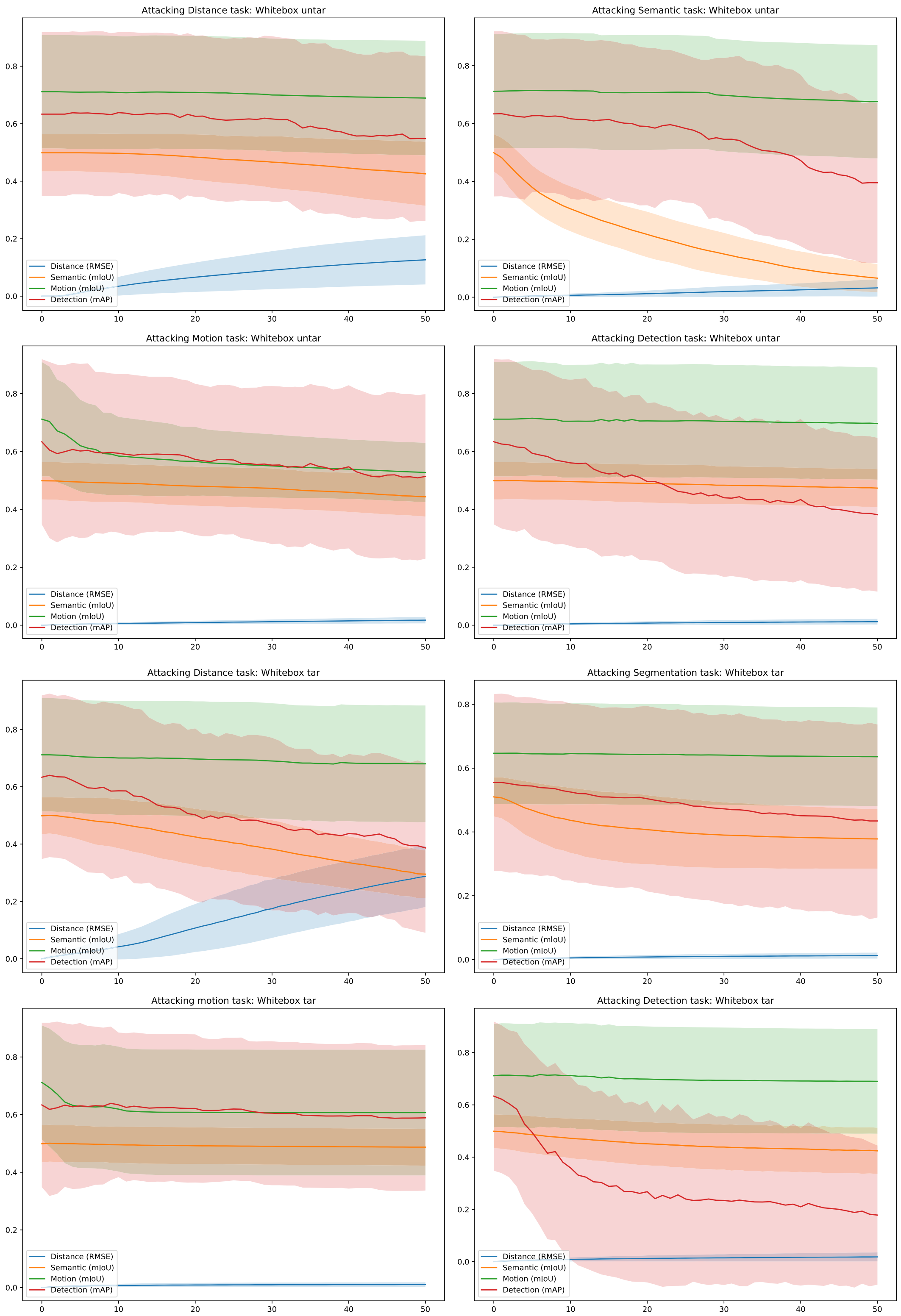


Figure 7.12 Performance comparison of **White-box** attacks across different tasks. The 1st and 2nd rows show untargeted attacks, 3rd and 4th rows show targeted attacks, and columns represent the tasks.

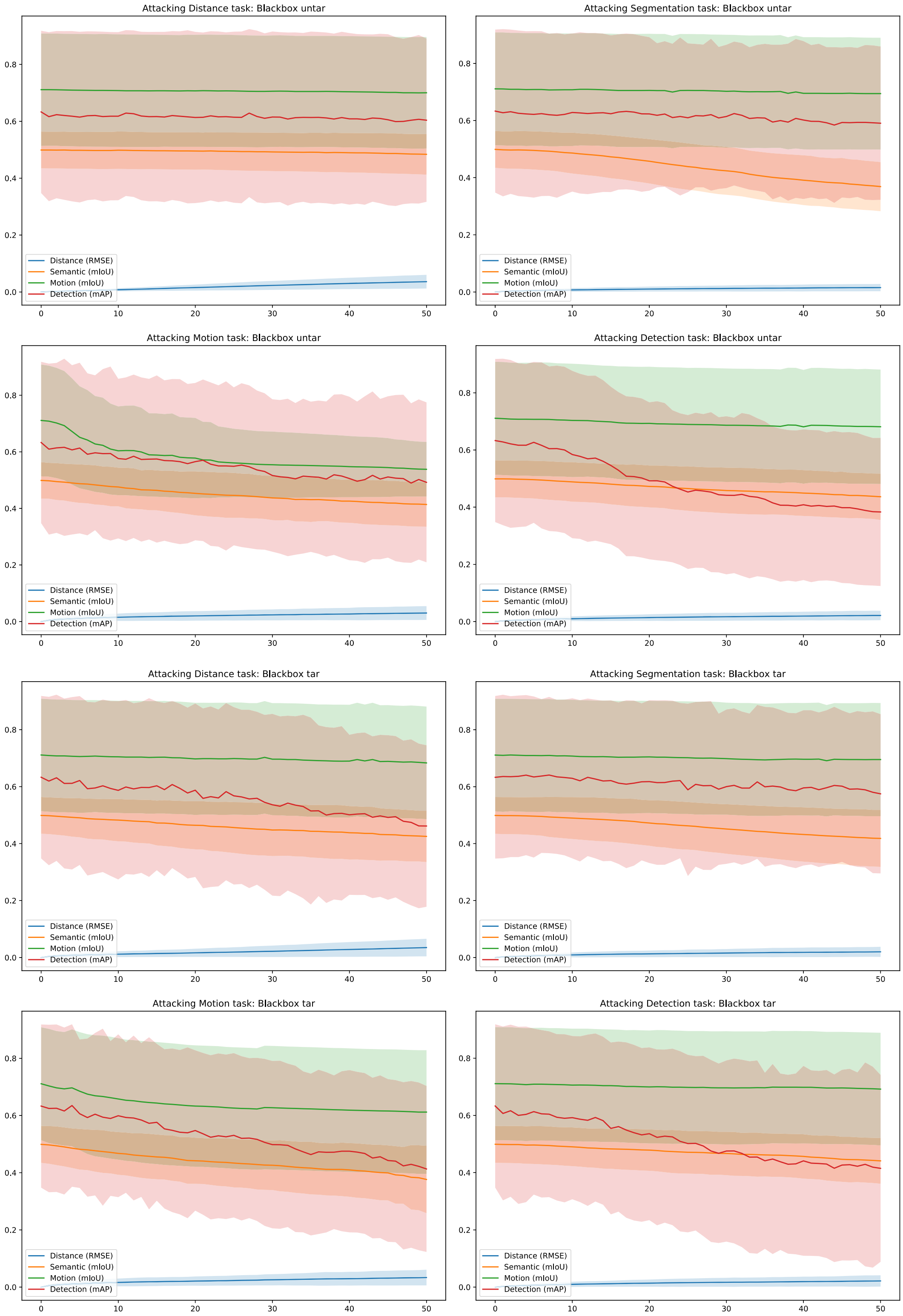


Figure 7.13 Performance comparison of **Black-box** attacks across different tasks. The 1st and 2nd rows show untargeted attacks, 3rd and 4th rows show the targeted attacks, and columns represent the attacked tasks.

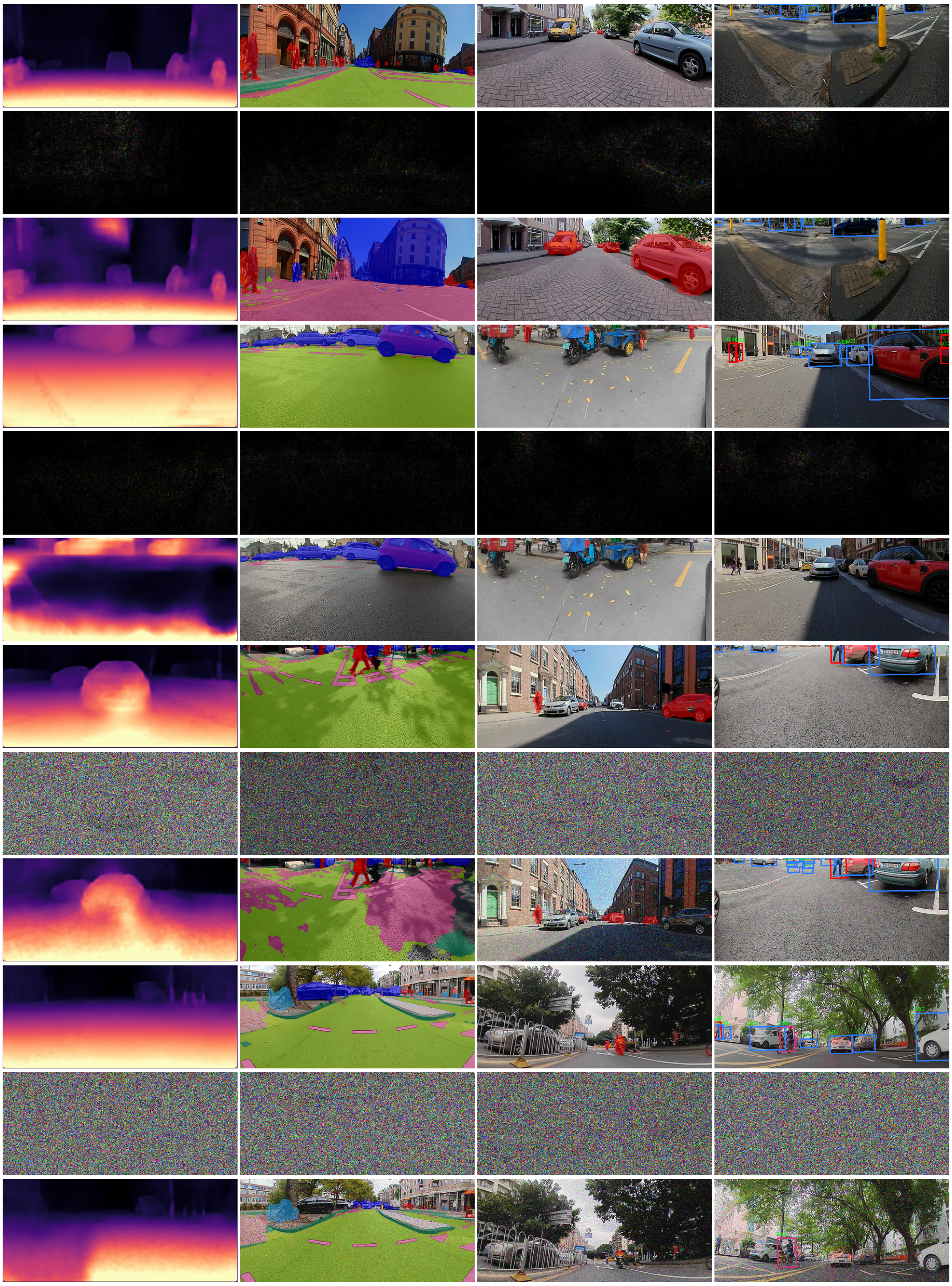


Figure 7.14 From Top to Bottom: **White box Untargeted, White box Targeted, Black box Untargeted, and Black box Targeted Attacks.** Within each group from top to bottom: Original results, adversarial perturbations, and the impacted results. For intuitive visualizations of the attacks, see `https://youtu.be/R3JUV41aiPY`.

Chapter 8

Discussion

Contents

This chapter discusses the contributions and the limitations of the approaches presented in this thesis.

8.1 Geometric Tasks

Depth estimation models may be learned in a supervised fashion on LiDAR distance measurements, such as KITTI [116]. In previous work, we followed this approach and demonstrated the possibility to estimate high-quality distance maps using LiDAR ground truth on fisheye images [10]. However, the limitation of this approach is that setting up the entire rig for such recordings is expensive and time-consuming, limiting the amount of data on which a model can be trained. Supervised learning remains a bottleneck for creating more intelligent generalized models which can do multiple tasks and acquire new skills without massive amounts of labeled data. To overcome this problem, we proposed *FisheyeDistanceNet* in **Chapter** 4, the first end-to-end self-supervised monocular scale-aware training framework. *FisheyeDistanceNet* uses CNNs on raw fisheye image sequences to regress a Euclidean distance map and provides a baseline for single frame Euclidean distance estimation.

Practically speaking, if we consider the supervised learning approach, it is improbable to label everything in the world. Suppose AI systems can gather a more profound, more nuanced knowledge of reality beyond what is specified in the training data set. In that case, they will be more valuable and eventually bring AI closer to human-level intelligence. We think that self-supervised learning is one of the most assuring means to develop such background knowledge and approximate common sense in AI systems. Self-supervised learning allows AI systems to learn from orders of magnitude more data, which is vital to identify and interpret patterns of more subtle, less common representations of the world.

8.1.1 Contributions

- A self-supervised training strategy that aims to infer a distance map from a sequence of distorted and unrectified raw fisheye images.

- A solution to the scale factor uncertainty with the bolster from ego-motion velocity allows outputting metric distance maps. This facilitates the map's practical use for self-driving cars.

- A novel combination of super-resolution networks and deformable convolution layers [417] to output high-resolution distance maps with sharp boundaries from a low-resolution input. Inspired by the super-resolution approach [419], we accurately resolve distances by replacing the deconvolution [428] and a naive nearest neighbor or bilinear upsampling. (see Figure 4.5: the self-supervised model, *FisheyeDistanceNet*, produces sharp, high quality distance and depth maps).

- We depict the importance of using backward sequences for training and construct a loss for these sequences. Moreover, a combination of filtering static pixels and an ego mask is employed. The incorporated bundle-adjustment framework [415] jointly optimizes distances and camera poses within a sequence by increasing the baseline and providing additional consistency constraints.

- A novel generic end-to-end self-supervised training pipeline to estimate monocular depth maps on raw distorted images for various camera models.

- Empirical evaluation of the approach on two diverse automotive datasets, namely KITTI and WoodScape.

- First demonstration of depth estimation results directly on unrectified KITTI sequences (see Figure 4.8: the scale-aware model, *UnRectDepthNet*, yields precise boundaries and fine-grained depth maps).

- State-of-the-art results on KITTI depth estimation among self-supervised methods.

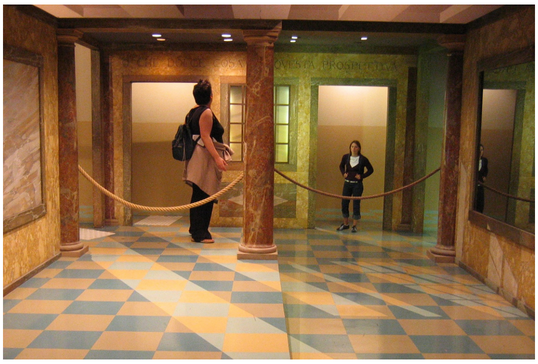


Figure 8.1 **Ames room**, a famous example of forced perspective, taken at Cité des Sciences, Paris [459].

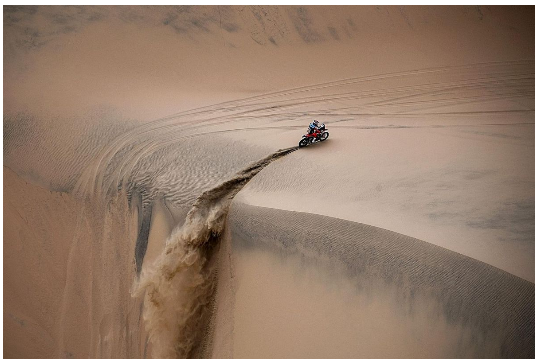


Figure 8.2 **Dakar 2019** photograph by Frank Fife [460].

8.1.2 Contextual Depth Limitations

It has been demonstrated that depth inference networks can estimate depth solely based on perspective and context. They were able to obtain reasonable scale-invariant quality measures with only one image [50, 53, 175, 457]. It has been deemed popular and convincing enough to prompt the development of a dedicated large-scale challenge [143]. Without a way to link the estimation to the real world, scale-invariant quality is not particularly interesting in the context of autonomous driving. Using human perception as a reference, it was demonstrated by McManus *et al.* [458] that forcing perspective on humans was well-known and studied, especially in the art for dramatic effect. *Ames room* is one of these methods. The apparent viewpoint is disputed by the person's height, as seen in Figure 8.1; the depth is thus different on the left and right, although it appears to be the same. This demonstrates that the projection of a point P is independent of its distance from the camera and the human eye's lack of robustness in determining depth from context while viewing a single image. We can infer that a single frame depth network will face the same constraints [406].

One could argue that these forced perspective counter-examples are not really realistic: if they were discovered unintentionally, the human would have been resistant to it. Figure 8.2 depicts a Dakar 2019 competitor. At first glance, an illusory cliff can be seen before realizing the image is a high-angle shot (that the human eye is not familiar with). This demonstrates that even realistic imagery can lead to a perplexing perspective. CNN's can learn depth from single images by focusing their attention on specific features, similar to how humans see. Because this process is entirely data-driven (no geometry is involved in deployment), biases in the training data significantly impact the accuracy of a monocular network. Van Dijk and de Croon [396] investigate this by taking a set of cues (relative position, apparent size, and others) and examine how they affect the depth maps estimated by CNNs trained on KITTI [11] as shown in Figure 8.3. In summary, the position vs. apparent size is as follows:

- Varying position and size **affects** depth estimation (the higher and smaller, the farther) as seen in the 1$^{\text{st}}$ row in Figure 8.3.
- Varying position only **affects** depth as well (the higher, the farther) as seen in the 2$^{\text{nd}}$ row.
- Varying size only **does not affect** estimated depth as seen in the 3$^{\text{rd}}$ row.

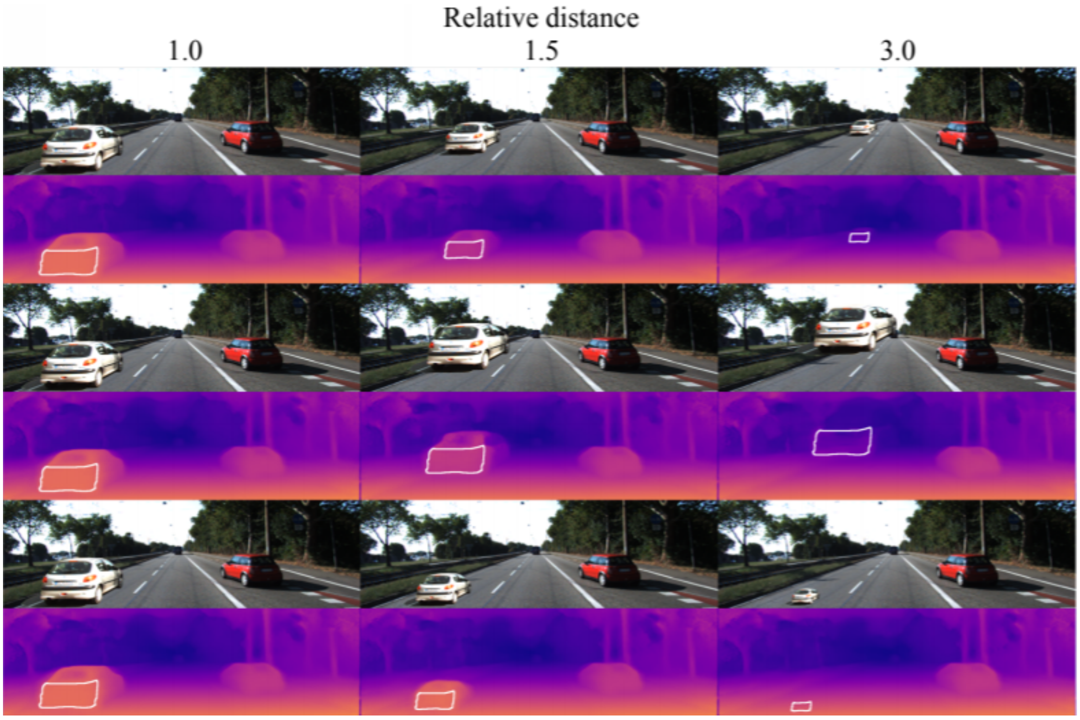


Figure 8.3 **Illustration pf depth estimation's effect on change in position vs. apparent size on a test image from KITTI.** The white car on the left is inserted into the image at 1.0 (left column), 1.5 (middle column), and 3.0 (right column), where 1.0 corresponds to the same scale and position at which the car was cropped from its original image. The position and scale of the car vary with distance in the top row; in the middle row, only the position changes while the scale remains constant, and in the bottom row, the scale varies while the position remains constant. A white outline indicates the measurement region from which the estimated distance is obtained in the disparity maps. Figure and part of the caption reproduced from [396].

These examples show that depth from context is not robust to new environments and that the depth error (even when scale-invariant) is not continuous with respect to appearance. It is also prone to changes in apparent camera mounting and object positions vs. apparent change in the size of the objects and if the image is cropped at a different aspect ratio for inference than the one used during training. Significant errors can even be found in the training set due to multi-stable perception [461]. As a result, a training set must be comprehensive because a minor change in appearances, such as a change in lighting or orientation, can completely change the outcome.

Furthermore, a depth estimator trained on this hypothetically may be required to remember the most probable perspective layout in a given image, implying that heavy memory tasks may be required. We can assume that a neural network dedicated to this task will have to be large and thus complex to be embedded on a mobile system [406].

8.1.3 Shortcomings of Self-Supervised Distance Estimation

SfM pipelines have a varying degree of artifacts. Still, there exist several problems that need to be addressed. In this section, we will discuss some of the limitations.

In Chapter 4, *SfM* relies on the assumption of a static world. A real-world scene typically includes both *static objects* such as walls, structures, and buildings, as well

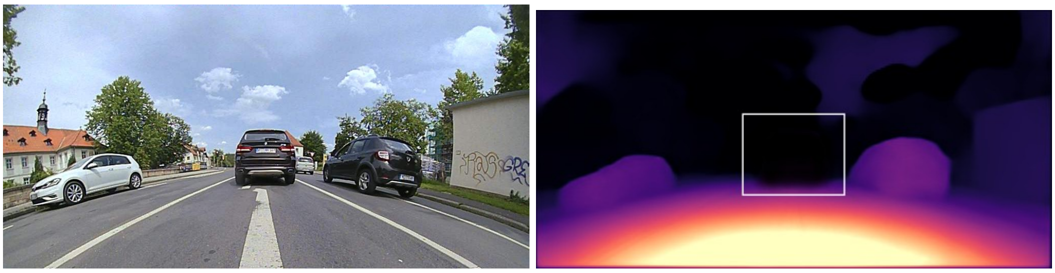


Figure 8.4 **Depiction of infinite distance due to dynamic objects** on FisheyeDistanceNet inducing holes during inference.

as *moving objects* such as persons and vehicles. We are assuming a static world hypothesis employing *SfM*, and it can only fulfill the static part.

What exactly does the static world assumption mean for the *SfM* framework?
Camera motion and depth structure are solely responsible for the movement of static objects in a frame. The depth structure and camera motion will thoroughly evaluate the projected 2D image motion between frames.
On the other hand, dynamic objects possess the characteristics of large displacement (*e.g.*, optical flow). They are caused by camera motion as well as actual object motion. This is not modeled in *SfM* frameworks.

8.1.4 Implications of the Used Loss Functions

Aside from violating the static world assumption, the system introduced a slew of new considerations.

- **Omission of object motion:** The object motion is ignored; when projecting the target pixels into the other camera view using the estimated pose, the reconstructed scene ignores motion in the source scene and is solely dependent on the estimated ego-motion. As a result, there is a mismatch between the target and predicted frames for pixels with motion. This means that even though the model correctly predicts the depth, it will still be penalized.

- **Depth efficiency is inextricably linked to the pose network performance:** Since the pose network is in charge of calculating the relative change in view, the performance of depth will suffer if the pose network underperforms.

- **The inevitable issue of predicting 'infinite depth' or holes:** It is mainly due to dynamic objects and a very particular case in which the object moves at the same speed as the camera (see Figure 8.4). The object will appear to be stationary relative to the camera. It can also occur when things are infinitely far away. Any movement of the object is deemed unobservable and appears stationary. Since the object is not moving, it can cause the depth network to predict an infinite depth, manifesting as holes.
  To reduce the impact of stationary pixels on the loss, we incorporated an auto masking strategy that ignores these pixels when calculating the loss between the source and target frames [55]. To further eliminate its impact on the loss, we tackled this issue in Chapter 5 with the help of semantic segmentation by employing the MTL approach.

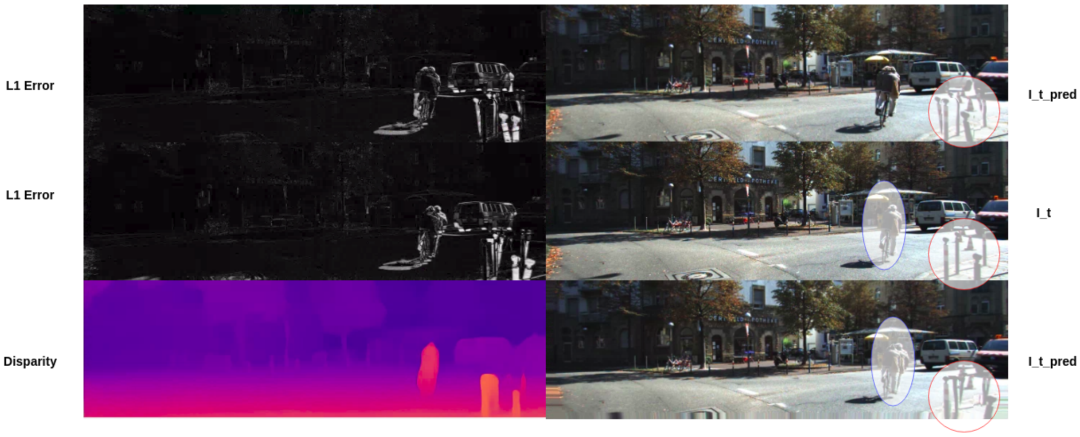


Figure 8.5 **Illustration of *SfM* framework's limitations.** It is observed that the model struggles near boundaries and thin elongated structures, as illustrated by the pixel-wise absolute difference between prediction and target image. A higher intensity in the L_1 error map indicates a higher absolute difference. Figure reproduced from [462].

8.1.5 Are there Better Choices than the Photometric Loss?

Appearance-based loss has been a reliable alternative to depth loss. However, there are several drawbacks to using image intensity as a performance indicator (see Figure 8.5).

- We cannot tell how big the world is from a 2D RGBD image. We lose scale information after the bilinear sampling stage since we compute the loss in 2D image space. In other words, several reconstructed images from the source view will result in the same correct target image. As a result, in 3D metric space, the learned depth will be scale uncertain but consistent in image space. To tackle this issue, we introduced scale awareness into the SfM framework 4.3.3 and computed direct depth/distance. In terms of KITTI Benchmark [11] results, several works [53, 55, 154, 363] depend on determining the median depth based on the ground truth to overcome the scale ambiguity.

- Another risk of scale ambiguity is the probability of depth collapsing. Wang *et al*. [177] demonstrated that as scale decreases, the same scene could be built with a smaller depth map until it degenerates to zero. Based on these findings, we introduced a normalization stage to prevent the mean depth from collapsing in Section 4.3.5.

- Even if the reconstructed image is not geometrically consistent, the loss can be satisfied. When depth is incorrectly predicted but pose is correct, or vice versa, the reconstructed image can still be well predicted in certain scenes. It is more common in texture-less regions such as the wall and the sky. The depth and pose networks are not penalized or compensated for incorrect predictions in this scenario.

- The value of the loss may be significant for differing views, such as when pixels are visible in the target frame but occluded or out of view in the source frames. There would be a mismatch in these regions when the points are re-projected. As a result, even though the depth is correctly estimated, the model will still be penalized.

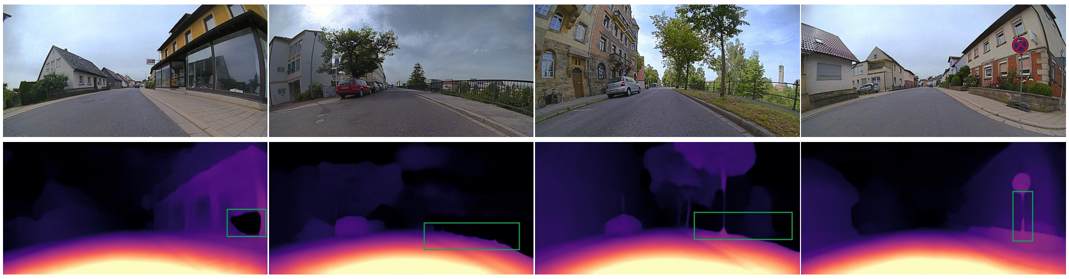


Figure 8.6 **Failure cases of FisheyeDistanceNet on the WoodScape.** The photometric loss fails to learn good distances for reflective regions which can be seen in the 1st figure. In the following figures shown above, the model fails to accurately delineate objects where boundaries are ambiguous.

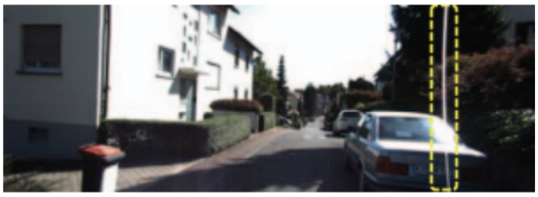


Figure 8.7 **An illustration of random lens error on the KITTI.** Figure reproduced from [462].

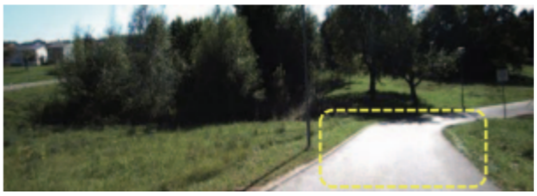


Figure 8.8 **An illustration of reflective surface on the KITTI.** Figure reproduced from [462].

- Reflective surfaces that violate the Lambertian assumption between views (see Figures 8.6, 8.7 and 8.8), and color-saturated images can cause mismatch and degrade the loss computation.

8.1.6 Impact on Chosen Distance Estimation Network

- Instead of preserving edges, networks often predict smooth depths along occlusion boundaries and depth discontinuities. A probable cause could be the lack of edge awareness explicitly defined and smooth surfaces dominating the loss. As a result, the bleeding edge effect [463] occurs, and the back-projected points are poorly defined, as shown in Figure 8.9.
- The distance map over consecutive frames, especially at the boundaries and thin structures in the video https://youtu.be/Sgq1WzoOmXg?t=27, causes a flickering effect since it is not geometrically consistent over time.

8.2 Geometry Meets Semantics

As heavily discussed in Chapter 4, today's state-of-the-art regarding self-supervision for depth sensing mostly relies on heuristics, themselves based on validation results on the KITTI dataset. A clear understanding of the theoretical aspect of self-supervision is needed to perform a truly robust training workflow. To overcome the shortcomings of the SfM framework for distance estimation, we analyzed the issues of the photometric loss function and the impact caused by the dynamic objects on the distance predictions, as discussed in the previous section. In Chapter 5, we dug deeper and took up the challenges posed by the SfM approach and the photometric losses.

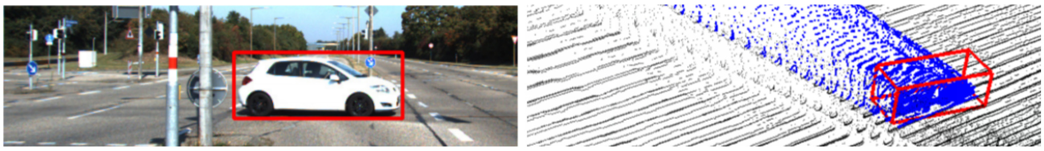


Figure 8.9 **Illustration of bleeding edge effect. Figure reproduced from [463].**

8.2.1 Contributions

- We introduce a novel architecture for the learning of self-supervised distance estimation synergized with semantic segmentation.
- We improve the self-supervised distance estimation by a general and robust loss function.
- We propose a solution for the dynamic object impact on self-supervised distance estimation by using semantic guidance. We show the approach's effectiveness on pinhole and fisheye camera datasets and present state-of-the-art results for both image types.
- We present a novel camera geometry adaptive multi-scale convolution to incorporate the camera parameters into the self-supervised distance estimation framework. We feed this camera geometry tensor (CGT) representation to the model as a generic way to adapt to new camera intrinsics.
- We create a training framework for self-supervised distance estimation, which jointly trains and infers images from multiple fisheye cameras and viewpoints.
- We demonstrate a single trained model for 23 fisheye cameras, which achieves the equivalent result as an individual specialized model that overfits a particular camera model.
- We present an improved version of the network architecture for multi-task learning of self-supervised distance estimation and semantic segmentation. We significantly improve upon the previous works explained in the previous chapters FisheyeDistanceNet [2] and UnrectDepthNet [3].
- We achieve state-of-the-art results on the WoodScape and KITTI datasets among monocular self-supervised depth estimation methods.

8.2.2 Exploring Diverse Modalities

Rethinking the Heuristic Loss Function

The difference between images is determined as the absolute difference between RGB values. At first, we aimed to improve the photometric loss by replacing the L_1 loss with a more robust loss function from [179] in the MTL framework of SynDistNet [4]. We replaced the loss with various types of general loss functions and achieved improved efficiency. The absolute loss is the same as maximizing the probability of a Laplacian distribution on RGB pixel values with a fixed scale. We substitute the general distribution for the fixed Laplacian distribution, keeping our scale constant but allowing the shape parameter α to vary.

Perspectives: Finally, as indicated by the SSIM [414] loss, a photometric loss that takes neighboring pixels into account may be advantageous to obtain more information than just color to determine whether two points match or not. Brox *et al.* [464] had already researched this and considered several features and descriptors to be matched in this work and concluded that including descriptor matching within the energy minimization process was beneficial, particularly for large displacement. The issue at the time was descriptors' lack of differentiability and sub-pixel accuracy. This problem could be easily adapted for photometric loss with convolution kernel descriptors, resulting in a feature matching loss as employed in Chapter 7.

In general, self-supervision should rely more on techniques developed for optical flow using the variational approach because occlusion issues and efficient smoothing and illumination robustness are not novel [406].

Leveraging Semantic Segmentation

In terms of representing structure in images, surface normal, disparity, and optical flow is very close to depth. The direction of the depth gradient can be interpreted as the surface normal. Ego-motion and object motion can naturally relate optical flow to depth. Depth has an inverse relationship with disparity. Although semantics does not have any direct relation, many use segmentation maps to learn finer depth maps. There have been many studies that have explored these mutually beneficial properties [462]. Our MTL network employed pixel-adaptive convolutions [242] to learn semantic-dependent representations that can better capture the aforementioned equivariance property compared to normal convolutions. We reduced the impact of dynamic objects on the photometric loss and inherently minimized the influence of 'infinite depth' or holes during inference. Our approach still lacked a complete solution *i.e.*, if a dynamic object did not belong to the semantic class labels, this would still have a negative impact on the SfM framework during training.

Model Advances

Recent work has shown that self-attention can assist as an essential building block for image recognition models. We adopted the self-attention modules from [8] and assessed their effectiveness for estimating distance/depth. Initially, we used a scalar-based self-attention module [430]. Later, we employed two types of self-attention. The first is pairwise self-attention, which is essentially a set operator and generalizes standard dot-product attention. The other choice is patchwise self-attention, which is more efficient than the standard convolution. Pairwise self-attention networks match or outperform their convolutional counterparts, and patchwise models outperform the convolutional baselines significantly. We investigate the robustness of learned representations and conclude that when used in conjunction with our Camera Geometry Tensor, self-attention networks can provide significant benefits in terms of model robustness and depth generalization.

Comparison between Convolution and Self-Attention

For the convolution operator, the fixed kernel weights are independent of the image's content. It does not adapt to the input content and can vary across channels, as shown in Table 8.1. Scalar attention when compared to convolution, the aggregated weights can vary across different locations depending on the image's content. The main drawback in the formulation is that it does not adapt the attention weights at the different channels. We can alleviate this to a certain extent by introducing multiple heads [431]; the number of heads is a small constant, and all channels within a head share scalar

Operation	*Content Adaptive*	*Channel Adaptive*
Convolution [465]	✗	✓
Scalar attention [429, 430, 431, 466]	✓	✗
Vector attention [8]	✓	✓

Table 8.1 **Comparison of convolution vs self-attention.** The content of the image is not adapted by the convolution. Scalar attention yields scalar weights that are constant along the channel dimension. Zhao *et al.*'s [8] self-attention modules compute attention weights that adapt across spatial dimensions and channels in an efficient manner.

weights. Zhao *et al.*'s [8] pairwise and patchwise modules can produce *vector* output. The vector can be processed and mapped to the appropriate dimensionality, which can also accept input from position encoding channels. Convolution is generalized by the patchwise family of operators while preserving parameter and FLOP efficiency.

8.3 Generalized Object Detection

8.3.1 Need for better 2D Object Representations

As discussed in Chapter 6, surround-view coverage is critical for low-speed maneuvering autonomous driving applications such as automated parking [60, 62]. Four surround-view fisheye cameras are typically part of this sensor suite, enabling a dense 360° near field perception. The wide field of view of the fisheye image comes with the side effect of strong radial distortion. A common practice is to rectify distortions in the image using a 4^{th} order polynomial model or a unified camera model [433]. However, undistortion comes with re-sampling distortion artifacts, especially at the periphery, a reduced FoV, and a non-rectangular image due to invalid pixels. Thus, we aimed to perform object detection on distorted fisheye images. Although semantic segmentation is an easier solution on fisheye images, object detection annotation costs are much lower [27].

8.3.2 Contributions

Chapter 6 aims to present a more detailed study of various techniques for fisheye object detection in autonomous driving scenes. The main contributions include:

- Exploration of seven different object representations for object detection on fisheye images.
- Design of novel representations for fisheye images, including the curved box and adaptive step polygon.
- Release of a dataset of 10,000 images with annotations for all the object representations.
- Implementation and empirical study of FisheyeYOLO baseline, which can output different representations.

8.3.3 Lack of Fisheye Object Detection Dataset

Our core objective was to present a more detailed study of various techniques for fisheye object detection in autonomous driving scenes. However, one of the main issues we faced in this research was the lack of a public dataset, particularly for autonomous driving scenarios. Henceforth, we compiled a 2D object detection dataset on fisheye images and released 10,000 images. We hope this encourages further research in this area leading to a mature object detection on undistorted fisheye images.

8.3.4 Limitations of Generalized Object Detection

Polygon representation is a better way to represent objects, but it does not improve the CNN ability to detect objects on fisheye. Since we use the YOLOv3 decoder, an area of improvement could be its average precision for medium and large objects. Compared to YOLOv2, mean average precision increased, and localization errors decreased. When comparing YOLO to RetinaNet [216], the YOLOv3's average precision does show a trade-off between speed and accuracy. By having a larger dataset, the accuracy of detecting objects with YOLOv3 can be made equal to the accuracy of RetinaNet, making it an excellent choice for models that can be trained with large datasets. YOLOv3 may not be suitable for use with niche models where large datasets are difficult to access [467].

8.4 Holistic 360° Scene Understanding

8.4.1 Contributions

This chapter demonstrates a multi-task perception model for the six essential perception tasks on unrectified fisheye images (shown in Figure 7.1). We discuss a full perception system building upon the tasks explained in the previous chapters, including depth estimation, pose estimation, semantic segmentation, and object detection. The contributions are as follows:

- We demonstrate the first real-time six-task model for surround-view fisheye camera perception.
- We propose novel design techniques, including the VarNorm task weighting.
- We design synergized decoders where various tasks help each other in addition to a shared encoder.
- We showcase a 6-task model on WoodScape and a 5-task model on KITTI and Cityscapes performing better than the single task baselines.
- We obtain state-of-the-art results for depth and pose estimation tasks on KITTI among monocular methods.

In the following sections, we briefly discuss some significant aspects of MTL as to why and when it works, when it does not, and what are the vital things to note when training an MTL framework.

Discussion on MTL: Though MTL is becoming more common, the 20-year-old hard parameter sharing paradigm remains prevalent in CNN-based MTL. Recent advancements in learning what to share, on the other hand, are encouraging. At the same time, our understanding of tasks – their similarity, relationship, hierarchy, and benefit for MTL – is minimal. We need to learn more about them better to understand MTL's generalization capabilities in DNNs.

8.4.2 Why Does MTL Work?

Even though an inductive bias acquired through multi-task learning seems intuitively possible, in order to better understand MTL, we must look at the processes that shape it. For all examples, we will assume that we have two related tasks, A and B, which rely on a common hidden layer representation F.

- **Implicit data augmentation:** MTL effectively improves the sample size for which we are training our model. Since all tasks are at least somewhat noisy, our goal when training a model on some task A is to learn a good representation for task A that ignores the data-dependent noise and generalizes well. Since different tasks produce different noise patterns, a model that learns two tasks simultaneously may learn a more general representation. Learning only task A risks overfitting to task A, while learning A and B together allows the model to achieve a better representation F by averaging the noise patterns [129].

- **Attention focusing:** It may be difficult for a model to distinguish between essential and irrelevant features when the task is very noisy or the data is limited and high-dimensional. MTL will assist the model in focusing its attention on the significant features, as other tasks will provide additional evidence for the importance or irrelevance of those features [129].

- **Eavesdropping:** The certain features G are simple to learn for one task B but challenging to learn for another. This could be due to A interacting with the features in a more nuanced way, or it could be due to other features impeding the model's ability to learn G. We can allow the model to eavesdrop through MTL, allowing it to learn G via task B. The simplest method is to use hints [468], which involves explicitly training the model to predict the most relevant features [129].

- **Regularization:** Finally, by adding an inductive bias, MTL serves as a regularizer. As a result, it reduces the chance of overfitting and the model's Rademacher complexity *i.e.*, its ability to fit random noise [129].

8.4.3 When MTL Works – And When It Does Not

Multi-task learning aims to use data from related tasks to improve the generalization performance of all tasks simultaneously. Shared hidden layers can transfer knowledge between related tasks in a neural network, reducing overfitting and improving learned latent representations, especially when task-specific training data is scarce. On the other hand, MTL can be detrimental to performance when the tasks being considered are not sufficiently related. Although multi-task performance may improve on average across all tasks, multi-task performance for some specific tasks may be worse than a single-task model. This drop in performance is referred to as **negative transfer**. Negative transfer is especially problematic when a subset of tasks is of primary interest, and the others are only used to improve representation learning. Liu *et al.* [469] proposes two hypotheses for why negative transfer might occur. (1) Because all tasks are diverse and unrelated to one another, and there is no suitable common latent representation, multi-task learning produces poor representations. (2) The

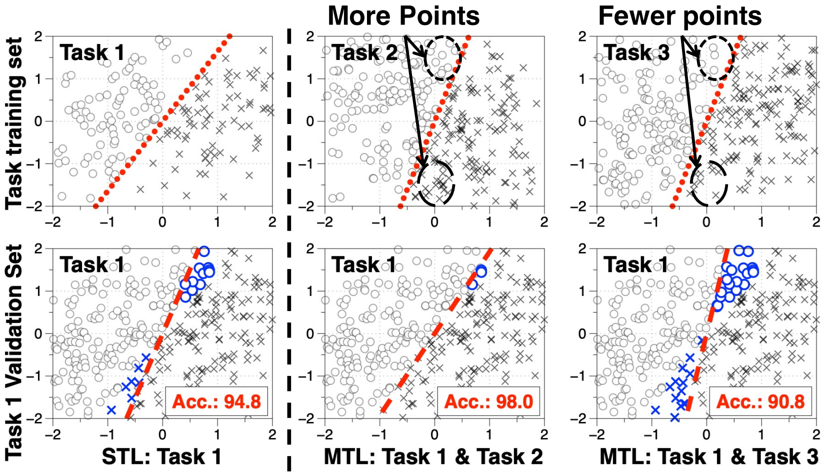


Figure 8.10 **Positive vs. Negative transfer is affected by the data – not just the model**. See lower right-vs-mid. Task 2 and 3 have the same model (dotted lines) but different data distributions. Notice the difference of data in circled areas. Figure and caption reproduced from [470].

training process is dominated by a single group of related tasks. The performance of those tasks improves as more related tasks are added, but tasks outside the dominant group suffer [469].

Wu *et al*. [470] showed that MTL, when applied to heterogeneous task data, can frequently produce suboptimal models. They investigate an architecture with a shared module for all tasks (*i.e.*, encoder) and a separate output module for each task (*i.e.*, decoder) to determine whether two tasks interfere constructively or destructively. The motivating observation is that, in addition to model similarity, which impacts the type of interference, task data similarity has a second-order effect after controlling model similarity. They consider three tasks with the same number of data samples, where tasks 2 and 3 have the same decision boundary but different data distributions (see Figure 8.10 for an illustration). They discover that training task 1 with task 2 or task 3 can either improve or degrade task 1's performance, depending on the amount of data contributing along the decision boundary! This finding demonstrates that by analyzing task interference and attributing the cause more precisely, we can measure the similarities of task data and models separately.

8.4.4 Key Components to Determine if MTL is better than STL

The three significant factors that can result in negative transfer and help us to determine if MTL is better than Single Task Learning (STL) are:

Model capacity: The shared module's capacity, *i.e.*, its output dimension, is critical because if the shared module is too large, there can be no interference (or transfer of knowledge) between tasks because each of them can be memorized in the shared module, resulting in zero training loss. There may be destructive interference if it is too small. As a general rule, according to [470] the shared module performs best when its capacity is less than the total capacity of the single-task models.

Task covariance: To determine the interference between different tasks, Wu *et al*. [470] uses a fine-grained concept called task covariance to measure how similar two tasks are. Task covariance quantifies the alignment of two task input data points along their primary axes. According to intuition, if the principal directions of two task input data are not well-aligned, feeding them into the shared module can result in suboptimal models. Wu *et al*. [470] proposed a covariance alignment algorithm that adds an alignment module between the task and the shared module to improve multi-task training to address this issue.

Optimization scheme: The order in which we optimize a multi-task learning neural network can also affect the interference between tasks. A common training strategy is to mix mini-batches of different task data at random. We can increase the task weight for an important task by duplicating its data. For further discussions on the optimization strategies employed in MTL refer Section 3.6.5.

Chapter 9

Conclusion

In this thesis, we explored geometric and segmentation tasks for a holistic real-time scene understanding of the environment *using cameras only*. The tremendous improvement in computer vision technology using deep learning-based models is the core reason why it is possible to drive on the road with camera-only perception. We build a near-field perception system that constitutes a *Level* 3 autonomous stack. We develop a single real-time multi-task CNN capable of running on embedded NVIDIA's Jetson AGX platform, covering the necessary modules for near-field sensing use cases such as parking or traffic jam assistance. The model's scope is not just limited to near-field sensing use cases. We can deploy it for navigation on highways and urban scenarios. The final model encodes the input raw fisheye stream into a single, high-dimensional tensor representing geometry, semantics, motion, and object detection. This tensor can be used to make driving decisions for an autonomous car.

In **Chapter** 2 and **Chapter** 3 we provided an in-detail basic intuition on the fisheye camera models and geometry, perception tasks and related works. We reasoned why depth estimation is a challenging task compared to the other semantic tasks and provided the reader the fundamental background in vision. In **Chapter** 4, we initially focused on solving one of the most challenging geometric problems *i.e.*, distance estimation on raw fisheye cameras using image-based reconstruction techniques, which is a challenging task, as the mapping between 2D images to 3D surfaces is an under-constrained problem. Depth/distance estimation is also an ill-posed problem as there could exist many possible incorrect depths per pixel, which can also recreate the novel view. We also trained an additional model to predict the relative rigid transformation between the video frames to achieve the visual odometry required to place the detected objects in a temporally consistent map.

We took up a more challenging problem of obtaining distance on fisheye cameras which undergoes large distortion compared to pinhole cameras. Most of the previous works worked on rectified pinhole cameras and obtained inverse-depth estimates in the *SfM* framework. The legacy continued to rely on the principle of rectification as the fundamental first step to get the view synthesis working in the first place. We broke this premise and showed that it is possible to obtain scale-aware direct distance estimates without rectifying fisheye and pinhole cameras on two diverse automotive datasets.

Following up on *FisheyeDistanceNet* and *UnRectDepthNet* we focused on improving the distance estimation further in **Chapter** 5 by using semantic segmentation in a multi-task learning set up. Our network in the *Syndistnet* MTL framework learned semantic-aware geometric representations that could disambiguate photometric ambiguities in a self-supervised learning *SfM* context. We integrated a generalized robust loss function, which improved performance significantly while removing the need for hyperparameter tuning with the reprojection loss. Finally, we reduced the artifacts caused by dynamic objects, violating static world assumptions, by using a semantic masking strategy. We significantly improved upon the RMSE of previous works on fisheye images in **Chapter** 4 by a 25% reduction in RMSE. As there is little work on fisheye cameras, we also evaluated the proposed method on KITTI using a pinhole model. We achieved state-of-the-art performance among self-supervised methods without requiring an external scale estimation.

We later extended this MTL framework from *Syndistnet* to multiple cameras and viewpoints. A 360° perception of scene geometry is essential for automated driving, notably for parking and urban driving scenarios. Typically, it is achieved using surround-view fisheye cameras, focusing on the near-field area around the vehicle. Most of the current depth estimation approaches focused on employing just a single camera, which cannot be straightforwardly generalized to multiple cameras. Besides, the depth estimation model needs to be deployed across different-sized car lines with varying camera geometries. Even in a single-car line, there are variations in intrinsics due to manufacturing tolerances. Deep learning models are sensitive to these changes, and it is practically infeasible to train and test on each camera variant. Thus, we introduced novel camera-geometry adaptive multi-scale convolutions, which utilize the camera parameters as a conditional input, enabling the model to generalize to unseen fisheye cameras. We incorporated this into our previous work *Syndistnet*.

Additionally, we improved the distance estimation by pairwise and patchwise vector-based self-attention encoder networks, further increasing the distance estimation's performance. We evaluated our approach on the Fisheye WoodScape surround-view dataset, significantly improving over previous approaches, *i.e.*, *FisheyeDistanceNet*, *UnRectDepthNet*, and *SynDistNet*. We also showed a generalization of our approach across different camera viewing angles and performed extensive experiments to support our contributions. To enable comparison with other approaches, we evaluated the front camera data on the KITTI dataset (pinhole camera images) and achieved state-of-the-art performance among self-supervised monocular methods. The following work series carried out on distance estimation was very influential from a product perspective to win next-generation projects and was influential in the academic community. It encouraged the researchers in the robotics community to deploy fisheye cameras to obtain point clouds from depth maps for SLAM by eliminating the need for rectification on the camera streams.

In **Chapter** 6, we focused on the localization aspect of an autonomous car using 2D object detection. The standard bounding box fails in fisheye cameras due to the strong radial distortion, particularly in the image's periphery. We explored better representations such as oriented bounding box, ellipse, and generic polygon representations for object detection on fisheye images. One of the primary use cases is to find a free parking slot for the autonomous car with these representations on fisheye cameras.

In **Chapter** 7, we achieved the goal of the thesis, *i.e.*, a holistic real-time scene understanding for the near-field perception of the environment *using cameras only* by creating *OmniDet*: A surround-view camera-based multi-task visual perception network for autonomous driving. This thesis shows that a distinct approach can be considered to the immense challenge of autonomous driving: one that does not rely on infrastructures such as high-definition-maps or extremely costly sensor payloads yet can perform complex driving tasks using cameras. To achieve real-world driving efficiency, these learned vision representations are very crucial. With this framework's help, we can jointly understand and reason about geometry, semantics, motion, localization, and soiling from a single deep learning model at 60 frames/second on embedded systems. The accuracy, robustness, and performance of these models are much higher than before. These models were deployed in real-time on an autonomous vehicle using NVIDIA' Jetson AGX, and we were able to demonstrate these models performing robustly. We evaluated the network on five tasks on KITTI and Cityscapes, achieving competitive results. We set a benchmark of these tasks on the fisheye WoodScape dataset. We also applied various adversarial attacks to understand the vulnerabilities of the perception network. For each perception task, white and black box attacks were performed for targeted and un-targeted scenarios.

Finally, in Chapter 8, we discussed the contributions and the shortcomings of the perception tasks solved in this thesis. We reasoned about the design choices and their impact employed in this work. For the task of depth estimation, we discussed the fundamental limitations of depth from context and structure. However, after conducting a thorough investigation into the potential drawbacks of depth from vision algorithms, we were able to identify several issues with depth networks that would need to be addressed before they could be used as a reliable source of information for parking scenarios for autonomous cars. We dug deep into the core of the *SfM* approach and its implication on the loss. We reasoned about the apparent failures due to the heuristic design of the photometric loss function and its impact on the network. To overcome the shortcomings of the *SfM* framework, we analyzed the issues put forth during our initial research and explored diverse modalities, including re-looking the heuristic loss function, leveraging semantic segmentation, model advances, and compared the standard convolution with self–attention modules. Coming to the task of 2D object detection, we discussed the need for better 2D representations on fisheye images and the limitations of our model. Finally, we discussed the critical aspects of an MTL approach to determine if there is any negative transfer.

9.1 Future Work

We have showcased state-of-the-art methods for distance and depth estimation tasks. There are many more avenues left unexplored, and we outline some of them here. The distance estimates are applied independently on each frame, indicating that adding temporal consistency [471] would likely improve results with the usage of recurrent [444] or a transformer model [472]. Additionally, assisting the model with depth hints [180] would improve the results and also improve the reconstruction loss function. It could also guide the network to learn better weights. Hu *et al.* [473] claimed that only a portion of the image is relevant for a network to estimate depth, in analogy with human vision. Further research in this direction is required to understand how a CNN can infer depth from single images. For collision avoidance, it is intriguing to understand how confident the model is about a particular inference. Learning the uncertainty [474] of the predicted distance maps would be of significant importance for practical applications such as autonomous driving. More generally, the distribution of possible values can be very beneficial in decision-making. Instance aware projection consistency [475] can be introduced to the view synthesis process to improve the training.

Learning dense semantic representations in an unsupervised fashion is a significant problem in computer vision. As of now, most of the semantic and motion segmentation tasks are supervised. However, we can not rely on having a supervised label for every possible image type we encounter in the real world. One of the most prominent challenges for a real-world computer vision system is learning a representation from limited data that generalizes to novel situations. It is especially true for autonomous driving – the variety of road scenes that exist is tremendously wide. There are many promising methods to learning robust representations. Self-supervision would be a better option as obtaining accurate, pixel-wise semantic and motion labels for every sample in a dataset is a labor-intensive process that costs significant amounts of money and time [476]. Gansbeke *et al.* [477] tries to perform unsupervised semantic segmentation by contrasting object mask proposals. Tosi *et al.* [304] employs a self-supervised optical flow to segment the dynamic objects and obtain a motion mask.

For vision to become truly ubiquitous with intelligent robotic decision making [478], with all the perception algorithms using cameras, the key takeaway would be online learning. An intriguing viewpoint developed throughout this thesis is the concept of a robust and evolving geometric and semantic sensing system based on vision. In other words, such a system does not require any active sensors and can improve while in use. For example, Casser *et al.* [189] discussed this, where it was used to overfit each example of the test set (with a reset of the network between examples). This technique has the potential to be extremely useful in the context of stealth off-road autonomous vehicles. Because of their stealthiness, no active depth sensors can be used, and depth must be calculated using cameras. Furthermore, off-road environments are typically more heterogeneous than on-road environments, and our robustness-oriented solution may be superior to a traditional *SfM* analytical algorithm or an evolutive single frame depth solution. We find it engaging to examine whether and how it would be possible to self-adapt the *OmniDet* framework online. A perception system would require determining a complex, plausibly high-dimensional state and needs to represent the geometry, semantics, motion, and localization. It would be intriguing to use this framework as a baseline and achieve end-to-end driving by combining the perception and decision-making algorithms.

To deploy the perception systems on an autonomous car would still involve several safety challenges. Development and validation of algorithms through mere brute force testing is not possible. We would need to quantify uncertainty, interpret saliency for decisions, understand intermediate representations, and reason with multiple sensors in real-time. Robustness is of higher priority than precision, *e.g.*, we do not need to know other cars' position to the nearest millimeter to enable safe driving. We solely care if we can detect the car or not and its rough spatial layout within the scene. The same is true for the algorithms we build; we care about robustness rather than millimeter-level accurate geometry from a LiDAR laser scanner [478]. It would also be of significant interest to improve the robustness by strengthening or equipping the MTL system to resist adversarial attacks [479]. In the future, it would be necessary to conduct physical attacks on mature systems to assess the vulnerabilities of these networks and attack multiple tasks jointly. Adversarial attacks and defenses are still challenging tasks and an active area of research, especially for autonomous driving applications with multi-task deep networks.

To conclude, over the past few years, computer vision technology has evolved to a point where it operates robustly in the wild. We are backing that ***camera-only perception*** *will drive the smart autonomous cars and robots of the future. We hope that this thesis encourages further research in building a unified perception model for autonomous driving.*

Bibliography

[1] V. Ravi Kumar, S. Yogamani, H. Rashed, G. Sitsu, C. Witt, I. Leang, S. Milz, and P. Mäder, "OmniDet: Surround View Cameras based Multi-task Visual Perception Network for Autonomous Driving," in *IEEE Robotics and Automation Letters (RA-L) + 2021 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 6, no. 2, 2021, pp. 2830–2837. vii, 32, 35, 36, 139

[2] V. Ravi Kumar, S. A. Hiremath, M. Bach, S. Milz, C. Witt, C. Pinard, S. Yogamani, and P. Mäder, "Fisheyedistancenet: Self-supervised scale-aware distance estimation using monocular fisheye camera for autonomous driving," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 574–581. vii, ix, x, 37, 67, 82, 84, 90, 95, 104, 105, 107, 109, 110, 111, 112, 140, 152, 153, 171

[3] V. Ravi Kumar, S. Yogamani, M. Bach, C. Witt, S. Milz, and P. Mäder, "UnRectDepthNet: Self-Supervised Monocular Depth Estimation using a Generic Framework for Handling Common Camera Distortion Models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2020, pp. 8177–8183. vii, ix, 10, 37, 68, 78, 90, 104, 107, 109, 110, 111, 112, 138, 152, 171

[4] V. Ravi Kumar, M. Klingner, S. Yogamani, S. Milz, T. Fingscheidt, and P. Mader, "Syndistnet: Self-supervised monocular fisheye camera distance estimation synergized with semantic segmentation for autonomous driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 61–71. vii, viii, 37, 90, 104, 105, 107, 109, 110, 112, 124, 141, 152, 171

[5] V. Ravi Kumar, M. Klingner, S. Yogamani, M. Bach, S. Milz, T. Fingscheidt, and P. Mäder, "SVDistNet: Self-Supervised Near-Field Distance Estimation on Surround View Fisheye Cameras," *IEEE Transactions on Intelligent Transportation Systems*, vol. abs/2104.04420, 2021. vii, 37, 90

[6] V. Ravi Kumar, S. Yogamani, S. Milz, and P. Mäder, "FisheyeDistanceNet++: Self-Supervised Fisheye Distance Estimation with Self-Attention, Robust Loss Function and Camera View Generalization," in *Electronic Imaging*. Society for Imaging Science and Technology, 2021. vii, viii

[7] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon, "3d packing for self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2485–2494. viii, 48, 62, 82, 90, 103, 104, 110, 111, 112, 152, 153

[8] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 076–10 085. viii, 95, 97, 99, 144, 149, 172, 173

[9] V. Ravi Kumar, S. Milz, C. Witt, M. Simon, K. Amende, J. Petzold, S. Yogamani, and T. Pech, "Near-field depth estimation using monocular fisheye camera:

A semi-supervised learning approach using sparse LiDAR data," in *CVPR Workshop*, vol. 7, 2018. ix, x

[10] V. Ravi Kumar, S. Milz, C. Witt, M. Simon, K. Amende, J. Petzold, S. Yogamani, and T. Pech, "Monocular fisheye camera depth estimation using sparse lidar supervision," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2853–2858. ix, x, 10, 37, 164

[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 3354–3361. ix, 6, 82, 166, 169

[12] S. Yogamani, C. Hughes, J. Horgan, G. Sistu, P. Varley, D. O'Dea, M. Uricár, S. Milz, M. Simon, K. Amende *et al.*, "Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, 2019, pp. 9308–9318. ix, 10, 11, 16, 23, 28, 29, 52, 55, 82, 126, 152

[13] M. Uricar, G. Sistu, H. Rashed, V. R. K. Vobecky, Antonin and, P. Krizek, F. Burger, and S. Yogamani, "Let's Get Dirty: GAN Based Data Augmentation for Camera Lens Soiling Detection in Autonomous Driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 766–775. x, 31

[14] H. Rashed, E. Mohamed, G. Sistu, V. Ravi Kumar, C. Eising, A. El-Sallab, and S. Yogamani, "Generalized Object Detection on Fisheye Cameras for Autonomous Driving: Dataset, Representations and Baseline," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 2272–2280. x, xii

[15] I. Sobh, A. Hamed, V. Ravi Kumar, and S. Yogamani, "Adversarial Attacks on Multi-task Visual Perception for Autonomous Driving," in *In-Review of the IEEE 24th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2021. xi, 139

[16] S. Houben, S. Abrecht, M. Akila, A. Bär, F. Brockherde, P. Feifel, T. Fingscheidt, S. S. Gannamaneni, S. E. Ghobadi, A. Hammam, A. Haselhoff, F. Hauser, C. Heinzemann, M. Hoffmann, N. Kapoor, F. Kappel, M. Klingner, J. Kronenberger, F. Küppers, J. Löhdefink, M. Mlynarski, M. Mock, F. Mualla, S. Pavlitskaya, M. Poretschkin, A. Pohl, V. R. Kumar, J. Rosenzweig, M. Rottmann, S. Rüping, T. Sämann, J. D. Schneider, E. Schulz, G. Schwalbe, J. Sicking, T. Srivastava, S. Varghese, M. Weber, S. Wirkert, T. Wirtz, and M. Woehrle, "Inspect, Understand, Overcome: A Survey of Practical Methods for AI Safety," *CoRR*, vol. abs/2104.14235, 2021. [Online]. Available: https://arxiv.org/abs/2104.14235 xi

[17] M. M. Dhananjaya, V. R. Kumar, and S. Yogamani, "Weather and Light Level Classification for Autonomous Driving: Dataset, Baseline and Active Learning," in *IEEE 24th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2021. xi

[18] A. Dahal, E. Golab, R. Garlapati, V. Ravi Kumar, and S. Yogamani, "RoadEdgeNet: Road Edge Detection System Using Surround View Camera Images," in *Electronic Imaging*. Society for Imaging Science and Technology, 2021. xii

[19] H. Rashed, E. Mohamed, V. R. K. Sistu, Ganesh and, C. Eising, A. El-Sallab, and S. Yogamani, "FisheyeYOLO: Object Detection on Fisheye Cameras for Autonomous Driving," *Machine Learning for Autonomous Driving NeurIPS 2020 Virtual Workshop*, 2020. xii

[20] A. Das, P. Křížek, G. Sistu, F. Bürger, S. Madasamy, M. Uřičář, V. Ravi Kumar, and S. Yogamani, "TiledSoilingNet: Tile-level Soiling Detection on Automotive Surround-view Cameras Using Coverage Metric," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6. xii

[21] M. Yahiaoui, H. Rashed, L. Mariotti, G. Sistu, I. Clancy, L. Yahiaoui, V. Ravi Kumar, and S. Yogamani, "FisheyeModNet: Moving object detection on Surround-View Cameras for Autonomous Driving," *arXiv preprint arXiv:1908.11789*, 2019. xii, 10, 57, 71

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic Differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017. xii, 81, 149

[23] "Road traffic injuries," Apr. 2021. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries 2

[24] S. O.-R. A. V. S. Committee *et al.*, "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems," *SAE Standard J*, vol. 3016, pp. 1–16, 2014. 2, 30

[25] "IF HUMANS HAVE TWO EYES SHOULDN'T CARS AS WELL?" Apr. 2021. [Online]. Available: https://www.foresightauto.com/if-humans-have-two-eyes-shouldnt-cars-as-well/ 3, 4

[26] M. Raaijmakers, "Towards environment perception for highly automated driving: with a case study on roundabouts," *article*, 2017. 3

[27] M. Siam, S. Elkerdawy, M. Jagersand, and S. Yogamani, "Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges," in *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*. IEEE, 2017, pp. 1–8. 3, 52, 173

[28] M. Siam, H. Mahgoub, M. Zahran, S. Yogamani, M. Jagersand, and A. El-Sallab, "Modnet: Motion and appearance based moving object detection network for autonomous driving," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2859–2864. 3, 57, 140, 149

[29] N. Tripathi, G. Sistu, and S. Yogamani, "Trained Trajectory based Automated Parking System using Visual SLAM," *arXiv preprint arXiv:2001.02161*, 2020. 3, 10

[30] S. Milz, G. Arbeiter, C. Witt, B. Abdallah, and S. Yogamani, "Visual slam for automated driving: Exploring the applications of deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018, pp. 247–257. 3

[31] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006. 3

[32] "New way for self-driving cars to 'see' objects," Apr. 2021. [Online]. Available: https://www.expresscomputer.in/news/new-way-for-self-driving-cars-to-see-objects/35451/ 3, 4

[33] "Computer Vision at Tesla," Apr. 2021. [Online]. Available: https://heartbeat.fritz.ai/computer-vision-at-tesla-cd5e88074376 4

[34] R. T. Mullapudi, W. R. Mark, N. Shazeer, and K. Fatahalian, "Hydranets: Specialized dynamic architectures for efficient inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8080–8089. 4

[35] "Parking Scenario," May 2021. [Online]. Available: https://www.car-engineer.com/ 5

[36] P. Viola, M. Jones *et al.*, "Robust real-time object detection," *International journal of computer vision*, vol. 4, no. 34-47, p. 4, 2001. 9

[37] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448. 9, 49, 50, 51

[38] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. 9, 27, 50

[39] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2117–2125. 9, 50

[40] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440. 9, 53, 54, 56

[41] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1520–1528. 9, 28, 53

[42] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017. 9, 40, 53, 55

[43] M. Uřičář, P. Křížek, G. Sistu, and S. Yogamani, "Soilingnet: Soiling detection on automotive surround-view cameras," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 67–72. 9, 10, 30, 58, 71, 140, 142, 149

[44] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*. Springer, 2003, pp. 363–370. 9

[45] A. Kundu, K. M. Krishna, and J. Sivaswamy, "Moving object detection by multi-view geometric techniques from a single camera mounted robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2009, pp. 4306–4312. 9, 57

[46] T.-H. Lin and C.-C. Wang, "Deep learning of spatio-temporal features with geometric-based moving point detection for motion segmentation," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3058–3065. 9, 57

[47] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2758–2766. 9

[48] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2462–2470. 9

[49] J. Vertens, A. Valada, and W. Burgard, "Smsnet: Semantic motion segmentation using deep convolutional neural networks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 582–589. 9, 57, 149

[50] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374. 9, 27, 39, 45, 81, 82, 110, 111, 112, 152, 166

[51] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2650–2658. 9, 38, 45, 53, 62, 63

[52] R. Garg, V. K. Bg, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 740–756. 9, 27, 47, 68, 70

[53] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1851–1858. 9, 27, 37, 38, 47, 68, 70, 72, 73, 76, 77, 78, 80, 81, 82, 84, 96, 110, 111, 112, 123, 152, 153, 166, 169

[54] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 270–279. 9, 38, 47, 48, 70, 75, 76, 78, 80, 123

[55] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3828–3838. 9, 27, 37, 48, 68, 70, 76, 77, 78, 80, 81, 82, 84, 90, 96, 110, 111, 112, 123, 152, 153, 168, 169

[56] R. Wood, "Fish-eye views, and vision under water," *Philosophical Magazine*, vol. 12, no. 6, pp. 159—-162, 1908. 10

[57] W. N. Bond, "A wide angle lens for cloud recording," *Philosophical Magazine*, vol. 44, no. 263, pp. 999—-1001, 1922. 10

[58] K. Miyamoto, "Fish eye lens," *Journal of the Optical Society of America*, vol. 54, no. 8, pp. 1060—-1061, 1964. 10, 13

[59] P. D. Thomas, *Conformal Projections in Geodesy and Cartography*. Washington: U.S. Government Printing Office, 1952. 10

[60] M. Heimberger, J. Horgan, C. Hughes, J. McDonald, and S. Yogamani, "Computer vision in automated parking systems: Design, implementation and challenges," *Image and Vision Computing*, vol. 68, pp. 88–101, 2017. 10, 29, 90, 173

[61] A. Dahal, J. Hossen, C. Sumanth, G. Sistu, K. Malhan, M. Amasha, and S. Yogamani, "DeepTrailerAssist: Deep Learning based trailer detection, tracking and articulation angle estimation on automotive rear-view camera," in *IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019*, 2019, pp. 2339–2346. 10

[62] J. Horgan, C. Hughes, J. McDonald, and S. Yogamani, "Vision-based driver assistance systems: Survey, taxonomy and advances," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 2032–2039. 10, 52, 173

[63] M. Drulea, I. Szakats, A. Vatavu, and S. Nedevschi, "Omnidirectional stereo vision using fisheye lenses," in *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2014, pp. 251–258. 10

[64] D. Caruso, J. Engel, and D. Cremers, "Large-scale direct slam for omnidirectional cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 141–148. 10

[65] Z. Gu, H. Liu, and G. Zhang, "Real-time indoor localization of service robots using fisheye camera and laser pointers," in *Proceedings of the International Conference on Robotics and Biomimetics (ROBIO)*, 2014. 10

[66] L. Gallagher, V. R. Kumar, S. Yogamani, and J. B. McDonald, "A hybrid sparse-dense monocular slam system for autonomous driving," in *2021 European Conference on Mobile Robots (ECMR)*. IEEE, 2021, pp. 1–8. 10

[67] S. Liu, P. Guo, L. Feng, and A. Yang, "Accurate and Robust Monocular SLAM with Omnidirectional Cameras," *Sensors*, vol. 19, no. 4494, 2019. 10

[68] S. Ji, Z. Qin, J. Shan, and M. Lu, "Panoramic SLAM from a multiple fisheye camera rig," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 159, pp. 169–183, 2020. 10

[69] H. Matsuki, L. von Stumberg, V. Usenko, J. Stückler, and D. Cremers, "Omnidirectional DSO: Direct Sparse Odometry With Fisheye Cameras," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, no. 4. IEEE, 2018, pp. 3693–3700. 10

[70] G. Sistu, I. Leang, and S. Yogamani, "Real-time joint object detection and semantic segmentation network for automated driving," *arXiv preprint arXiv:1901.03912*, 2019. 10

[71] J. Zhu, J. Zhu, X. Wan, and C. Xu, "Downside Hemisphere Object Detection and Localization of MAV by Fisheye Camera," in *Proceedings of the International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018. 10

[72] Y. Ye, K. Yang, K. Xiang, J. Wang, and K. Wang, "Universal Semantic Segmentation for Fisheye Urban Driving Images," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020, pp. 648–655. 10

[73] M. Uricár, J. Ulicny, G. Sistu, H. Rashed, P. Krizek, D. Hurych, A. Vobecky, and S. Yogamani, "Desoiling dataset: Restoring soiled areas on automotive fisheye cameras," in *IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019*. IEEE, 2019, pp. 4273–4279. 10, 30, 58

[74] L. Meng, T. Hirayama, and S. Oyanagi, "Underwater-drone with panoramic camera for automatic fish recognition based on deep learning," *Ieee Access*, vol. 6, pp. 17 880–17 886, 2018. 10

[75] K. Qiu, T. Liu, and S. Shen, "Model-based global localization for aerial robots using edge alignment," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, no. 3. IEEE, 2017, pp. 1256–1263. 10

[76] N. Zioulis, A. Karakottas, D. Zarpalas, and P. Daras, "Omnidepth: Dense depth estimation for indoors spherical panoramas," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 448–465. 10

[77] "Old church, taken using a fisheye lens in Paris," Mar. 2021. [Online]. Available: https://depositphotos.com/38262039/stock-photo-old-church-taken-fisheye-lens.html 11

[78] A. Basu and S. Licardie, "Alternative models for fish-eye lenses," *Pattern Recognition Letters*, vol. 16, no. 4, pp. 433–441, 1995. 11, 16, 19

[79] F. Devernay and O. Faugeras, "Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured enviroments," *Machine Vision and Applications*, vol. 13, pp. 14—-24, 2001. 11, 17

[80] J. Kannala and S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 28, no. 8, pp. 1335–1340, 2006. 11, 16, 126

[81] C. Geyer and K. Daniilidis, "A unifying theory for central panoramic systems and practical applications," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2000. 11, 18, 19

[82] B. Khomutenko, G. Garcia, and P. Martinet, "An enhanced unified camera model," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 137–144, 2015. 11, 18, 19, 52, 102, 126

[83] V. Usenko, N. Demmel, and D. Cremers, "The double sphere camera model," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 552–560. 11, 17, 18, 20, 102

[84] A. W. Fitzgibbon, "Simultaneous linear estimation of multiple view geometry and lens distortion," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1. IEEE Computer Society, 2001, pp. 125–132. 11, 16

[85] D. Schneider, E. Schwalbe, and H.-G. Maas, "Validation of geometric models for fisheye lenses," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 64, no. 3, pp. 259–266, 2009. 13

[86] C. Hughes, P. Denny, E. Jones, and M. Glavin, "Accuracy of fish-eye lens models," *Applied Optics*, vol. 49, no. 17, pp. 3338–3347, 2010. 13, 21

[87] A. E. Conrady, "Decentred lens-systems," *Monthly notices of the royal astronomical society*, vol. 79, no. 5, pp. 384–390, 1919. 13, 16

[88] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000. 13, 16

[89] D. Kim, J. Park, and J. Paik, "Extended fisheye lens model for practical geometric correction and image enhancement," *Optics Letters*, vol. 39, no. 21, pp. 6261–6264, 2014. 15

[90] D. C. Brown, "Alternative models for fish-eye lenses," *Photogrammetric Engineering*, vol. 32, no. 2, pp. 444–462, 1966. 16

[91] MATLAB, *(R2021a)*. Natick, Massachusetts: The MathWorks Inc., 2021. 16

[92] X. Ying, Z. Hu, and H. Zha, "Fisheye Lenses Calibration Using Straight-Line Spherical Perspective Projection Constraint," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2006. 16

[93] H. Wildenauer and B. Micusík, "Closed form solution for radial distortion estimation from a single vanishing point," in *Proceedings of the British Machine Vision Conference (BMVC)*, vol. 1, 2013, p. 2. 17

[94] M. Antunes, J. P. Barreto, D. Aouada, and B. Ottersten, "Unsupervised vanishing point detection and camera calibration from a single manhattan image with radial distortion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4288–4296. 17

[95] F. Bukhari and M. N. Dailey, "Automatic radial distortion estimation from a single image," *Journal of mathematical imaging and vision*, vol. 45, no. 1, pp. 31–45, 2013. 17

[96] J. Courbon, Y. Mezouar, and P. Martinet, "Evaluation of the unified model of the sphere for fisheye cameras in robotic applications," *Advanced Robotics*, vol. 26, no. 8-9, pp. 947–967, 2012. 17

[97] C. Hughes, P. Denny, M. Glavin, and E. Jones, "Equidistant fish-eye calibration and rectification by vanishing point extraction," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 32, no. 12, pp. 2289–2296, 2010. 17

[98] X. Ying and Z. Hu, "Can We Consider Central Catadioptric Cameras and Fisheye Cameras within a Unified Imaging Model," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2004, pp. 442–455. 18

[99] J. Courbon, Y. Mezouar, L. Eckt, and P. Martinet, "A generic fisheye camera model for robotic applications," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2007, pp. 1683–1688. 18

[100] J. Courbon, Y. Mezouar, and P. Martinet, "Evaluation of the Unified Model of the Sphere for Fisheye Cameras in Robotic Applications," *Advanced Robotics*, vol. 26, no. 8-9, pp. 947–967, 2012. 18

[101] H. Bakstein and T. Pajdla, "Panoramic mosaicing with a 180/spl deg/field of view lens," in *Proceedings of the IEEE Workshop on Omnidirectional Vision 2002. Held in conjunction with ECCV'02*. IEEE, 2002, pp. 60–67. 19

[102] J. Perš and S. Kovacic, "Nonparametric, model-based radial lens distortion correction using tilted camera assumption," in *Proceedings of the Computer Vision Winter Workshop*, vol. 1, 2002, pp. pp–286. 19

[103] G. Klančar, M. Kristan, and R. Karba, "Wide-angle camera distortions and non-uniform illumination in mobile robot tracking," *Robotics and Autonomous Systems*, vol. 46, no. 2, pp. 125–133, 2004. 19

[104] X. Mei, S. Yang, J. Rong, X. Ying, S. Huang, and H. Zha, "Radial lens distortion correction using cascaded one-parameter division model," in *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2015, pp. 3615–3619. 19

[105] A. L. Laubscher, "A basic investigation of perspective map projections," Thesis, The Ohio State University, 1965. 19

[106] J. P. Snyder, "Map Projections: A Working Manual," *US Geological Survey Professional Paper*, vol. 1395, 1987. 19

[107] C. Hughes, P. Denny, E. Jones, and M. Glavin, "Accuracy of fish-eye lens models," *Applied optics*, vol. 49, no. 17, pp. 3338–3347, 2010. 22, 102

[108] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015. 23, 110, 153

[109] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 834–849. 23

[110] "What Are 3D Reconstruction Models?" May 2021. [Online]. Available: https://moneyinc.com/3d-reconstruction-models/ 23

[111] "The Science of Vision: An Introduction," May 2021. [Online]. Available: https://mind.ilstu.edu/curriculum/vision%5Fscience%5Fintro/vision%5Fscience%5Fintro.html 23, 24

[112] "Depth Estimation: Basics and Intuition," Apr. 2021. [Online]. Available: https://towardsdatascience.com/depth-estimation-1-basics-and-intuition-86f2c9538cd1 24, 26, 68, 69, 70

[113] "How do people with depth perception issues drive a car and stop in traffic by providing adequate space and not crashing the car in front of theirs?" May 2021. [Online]. Available: https://www.quora.com/How-do-people-with-depth-perception-issues-drive-a-car-and-stop-in-traffic-by-providing-adequate-space-and-not-crashing-the-car-in-front-of-theirs 24

[114] F. L. Kooi and A. Toet, "Visual comfort of binocular and 3D displays," *Displays*, vol. 25, no. 2-3, pp. 99–108, 2004. 25

[115] J. S. Gardner, J. L. Austerweil, and S. E. Palmer, "Vertical position as a cue to pictorial depth: Height in the picture plane versus distance to the horizon," *Attention, Perception, & Psychophysics*, vol. 72, no. 2, pp. 445–453, 2010. 25

[116] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013. 25, 69, 71, 76, 82, 110, 125, 152, 153, 164

[117] "Motion Parallax in Psychology: Definition and Explanation," May 2021. [Online]. Available: https://study.com/academy/lesson/motion-parallax-in-psychology-definition-lesson-quiz.html 26

[118] "Retinal Disparity," May 2021. [Online]. Available: https://study.com/cimages/multimages/16/retinal%5Fdisparity%5Fview.png 26

[119] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010. 26

[120] M. Okutomi and T. Kanade, "A multiple-baseline stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 15, no. 4, pp. 353–363, 1993. 27

[121] Y. Boykov, O. Veksler, and R. Zabih, "A variable window approach to early vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 20, no. 12, pp. 1283–1294, 1998. 27

[122] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," *International Journal of Computer Vision*, vol. 35, no. 3, pp. 269–293, 1999. 27

[123] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1, pp. 7–42, 2002. 27

[124] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," *Artificial intelligence*, vol. 78, no. 1-2, pp. 87–119, 1995. 27

[125] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 21–37. 27, 50

[126] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 39, no. 6, pp. 1137–1149, 2017. 27, 50, 51

[127] "A 2021 guide to Semantic Segmentation," May 2021. [Online]. Available: https://nanonets.com/blog/semantic-image-segmentation-2020/ 28

[128] "CS224d: Deep Learning for Natural Language Processing," May 2021. [Online]. Available: http://cs224d.stanford.edu/index.html 28

[129] "An Overview of Multi-Task Learning in Deep Neural Networks," May 2021. [Online]. Available: https://ruder.io/multi-task/ 32, 33, 34, 175

[130] R. Caruana, "Multitask Learning. Autonomous Agents and Multi-Agent Systems," *Machine Learning*, p. 41–75, 1998. 33, 59

[131] R. Caruana, "Multitask Learning: A Knowledge-Based Source of Inductive Bias, ser," *Proceedings of the 10th International Conference on Machine Learning (ICML)*, 1993. 34

[132] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7482–7491. 34, 60, 62, 63, 94, 148, 149

[133] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *International Conference on Machine Learning*. Pmlr, 2018, pp. 794–803. 34, 60, 63, 148, 149

[134] O. Sener and V. Koltun, "Multi-Task Learning as Multi-Objective Optimization," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS), 2018*, 2018, pp. 525–536. 34, 60

[135] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Fast scene understanding for autonomous driving," *arXiv preprint arXiv:1708.02550*, 2017. 34, 60

[136] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1013–1020. 34, 60, 62, 139

[137] J. Baxter, "A Bayesian/information theoretic model of learning to learn via multiple task sampling," *Machine learning*, vol. 28, no. 1, pp. 7–39, 1997. 34

[138] I. Kokkinos, "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6129–6138. 34, 149

[139] L. Duong, T. Cohn, S. Bird, and P. Cook, "Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser," in *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers)*, 2015, pp. 845–850. 34

[140] Y. Yang and T. M. Hospedales, "Trace norm regularised deep multi-task learning," in *5th International Conference on Learning Representations, ICLR, Workshop Track Proceedings*. OpenReview.net, 2017. 34

[141] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3994–4003. 34, 60, 61

[142] Who, "Global status report on road safety 2018, "World Health Organization," https://apps.who.int/iris/bitstream/handle/10665/276462/9789241565684-eng.pdf, [Accessed October-2020]. 34

[143] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *2017 international conference on 3D Vision (3DV)*. IEEE, 2017, pp. 11–20. 38, 82, 111, 112, 152, 166

[144] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3061–3070. 38, 56, 57

[145] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2. IEEE, 2005, pp. 807–814. 38

[146] A. Saxena, S. H. Chung, A. Y. Ng *et al.*, "Learning depth from single monocular images," in *Nips*, vol. 18, 2005, pp. 1–8. 39

[147] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 824–840, 2008. 39

[148] I. Ulku and E. Akagunduz, "A survey on deep learning-based architectures for semantic segmentation on 2d images," *arXiv preprint arXiv:1912.10230*, 2019. 40, 53, 54

[149] "Intersection over Union (IoU) for object detection," May 2021. [Online]. Available: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection 41

[150] "Understanding the mAP Evaluation Metric for Object Detection," May 2021. [Online]. Available: https://medium.com/\spacefactor\@m{}timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3 41, 42

[151] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010. 42, 52

[152] "Metrics for object detection," May 2021. [Online]. Available: https://github.com/rafaelpadilla/Object-Detection-Metrics 42

[153] A. Moreau, M. Mancas, and T. Dutoit, "Depth prediction from 2D images: A taxonomy and an evaluation study," *Image Vis. Comput.*, vol. 93, p. 103825, 2020. 45, 47, 48

[154] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5667–5675. 45, 47, 48, 76, 81, 82, 110, 111, 112, 153, 169

[155] U. Franke, D. Gavrila, S. Gorzig, F. Lindner, F. Puetzold, and C. Wohler, "Autonomous driving goes downtown," *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 6, pp. 40–48, 1998. 45

[156] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox, "DeMoN: Depth and Motion Network for Learning Monocular Stereo," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE Computer Society, 2017, pp. 5622–5631. 45

[157] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki, "SfM-Net: Learning of Structure and Motion from Video," *ArXiv*, vol. abs/1704.07804, 2017. 46, 62, 76, 77

[158] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2024–2039, 2015. 46

[159] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5162–5170. 46

[160] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data,” in *Proceedings of the Eighteenth International Conference on Machine Learning (ICML*. Morgan Kaufmann, 2001, pp. 282–289. 46

[161] E. Shelhamer, J. T. Barron, and T. Darrell, “Scene intrinsics and depth from a single image,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 37–44. 46

[162] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, “Deeper depth prediction with fully convolutional residual networks,” in *2016 Fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 239–248. 46

[163] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. 46, 50, 54, 55, 79, 97, 99

[164] W. Chen, Z. Fu, D. Yang, and J. Deng, “Single-Image Depth Perception in the Wild,” in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, 2016, pp. 730–738. 46

[165] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *arXiv preprint arXiv:1406.2661*, 2014. 46, 58, 66

[166] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, “Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks,” in *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2017, pp. 5907–5915. 46

[167] Y. Hong, U. Hwang, J. Yoo, and S. Yoon, “How Generative Adversarial Networks and Their Variants Work: An Overview,” *ACM Comput. Survey*, vol. 52, no. 1, pp. 10:1–10:43, 2019. 46

[168] X. Huang, Y. Li, O. Poursaeed, J. E. Hopcroft, and S. J. Belongie, “Stacked Generative Adversarial Networks,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE Computer Society, 2017, pp. 1866–1875. 46

[169] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *arXiv preprint arXiv:1411.1784*, 2014. 46

[170] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2017, pp. 2223–2232. 46

[171] T. Feng and D. Gu, “Sganvo: Unsupervised deep visual odometry and depth estimation with stacked generative adversarial networks,” in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4, no. 4. IEEE, 2019, pp. 4431–4437. 46

[172] H. Jung, Y. Kim, D. Min, C. Oh, and K. Sohn, “Depth prediction from a single image with conditional adversarial networks,” in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 1717–1721. 46

[173] K. Gwn Lore, K. Reddy, M. Giering, and E. A. Bernal, “Generative adversarial networks for depth map estimation from RGB video,” in *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2018, pp. 1177–1185. 46

[174] C. Zhao, Q. Sun, C. Zhang, Y. Tang, and F. Qian, "Monocular depth estimation based on deep learning: An overview," *Science China Technological Sciences*, pp. 1–16, 2020. 46

[175] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2002–2011. 47, 166

[176] F. Zhang, V. Prisacariu, R. Yang, and P. H. Torr, "Ga-net: Guided aggregation net for end-to-end stereo matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 185–194. 47

[177] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey, "Learning depth from monocular videos using direct methods," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2022–2030. 47, 48, 77, 80, 81, 82, 110, 111, 112, 123, 153, 169

[178] F. Aleotti, F. Tosi, M. Poggi, and S. Mattoccia, "Generative adversarial networks for unsupervised monocular depth prediction," in *Computer Vision - ECCV 2018 Workshops, 2018, Proceedings, Part I*, vol. 11129. Springer, 2018, pp. 337–354. 48

[179] J. T. Barron, "A general and adaptive robust loss function," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4331–4339. 48, 90, 92, 93, 108, 111, 141, 171

[180] J. Watson, M. Firman, G. J. Brostow, and D. Turmukhambetov, "Self-supervised monocular depth hints," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2162–2171. 48, 180

[181] C. Shu, K. Yu, Z. Duan, and K. Yang, "Feature-metric loss for self-supervised learning of depth and egomotion," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 572–588. 48, 109, 110, 112, 141, 142, 144, 152

[182] A. CS Kumar, S. M. Bhandarkar, and M. Prasad, "Monocular depth prediction using generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018, pp. 300–308. 48, 112

[183] A. Pilzer, D. Xu, M. Puscas, E. Ricci, and N. Sebe, "Unsupervised adversarial depth estimation using cycled generative networks," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 587–595. 48

[184] F. Tosi, F. Aleotti, M. Poggi, and S. Mattoccia, "Learning monocular depth estimation infusing traditional stereo knowledge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9799–9809. 48

[185] B. Bozorgtabar, M. S. Rad, D. Mahapatra, and J.-P. Thiran, "Syndemo: Synergistic deep feature alignment for joint learning of depth and ego-motion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4210–4219. 48

[186] J. Zhou, Y. Wang, K. Qin, and W. Zeng, “Unsupervised high-resolution depth learning from videos with dual networks,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 6872–6881. 48, 112

[187] A. Pilzer, S. Lathuiliere, N. Sebe, and E. Ricci, “Refine and distill: Exploiting cycle-inconsistency and knowledge distillation for unsupervised monocular depth estimation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9768–9777. 48

[188] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, “Unsupervised monocular depth and ego-motion learning with structure and semantics,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019*, 2019, pp. 381–388. 48, 62

[189] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, “Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8001–8008. 48, 62, 76, 82, 90, 104, 109, 110, 111, 112, 152, 153, 181

[190] R. Wang, S. M. Pizer, and J.-M. Frahm, “Recurrent neural network for (un-) supervised learning of monocular video visual odometry and depth,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5555–5564. 48

[191] H. Zhang, C. Shen, Y. Li, Y. Cao, Y. Liu, and Y. Yan, “Exploiting temporal consistency for real-time video depth estimation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1725–1734. 48

[192] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, “Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8977–8986. 48, 112

[193] F.-E. Wang, Y.-H. Yeh, M. Sun, W.-C. Chiu, and Y.-H. Tsai, “Bifuse: Monocular 360 depth estimation via bi-projection fusion,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 462–471. 48

[194] L. Jin, Y. Xu, J. Zheng, J. Zhang, R. Tang, S. Xu, J. Yu, and S. Gao, “Geometric structure based and regularized depth estimation from 360 indoor imagery,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 889–898. 48

[195] C. Ma, L. Shi, H. Huang, and M. Yan, “3d reconstruction from full-view fisheye camera,” *arXiv preprint arXiv:1506.06273*, 2015. 48

[196] S. Pathak, A. Moro, A. Yamashita, and H. Asama, “Dense 3D reconstruction from two spherical images via optical flow-based equirectangular epipolar rectification,” in *2016 IEEE International Conference on Imaging Systems and Techniques (IST)*. IEEE, 2016, pp. 140–145. 48

[197] S. Li and K. Fukumori, “Spherical stereo for the construction of immersive VR environment,” in *IEEE Proceedings. VR 2005. Virtual Reality, 2005.*, 2005, pp. 217–222. 48

[198] G. Sistu, I. Leang, S. Chennupati, S. Yogamani, C. Hughes, S. Milz, and S. Rawashdeh, "Neurall: Towards a unified visual perception model for automated driving," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 796–803. 48, 60, 62, 139

[199] S. Chennupati, G. Sistu, S. Yogamani, and S. Rawashdeh, "AuxNet: Auxiliary Tasks Enhanced Semantic Segmentation for Automated Driving," in *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, 2019, pp. 645–652. 48, 62

[200] R. Hartley and A. Zisserman, "Multiple view geometry in computer vision (cambridge university, 2003)," *C1 C3*, vol. 2, 2003. 48, 71

[201] Y. Furukawa and C. Hernández, "Multi-view stereo: A tutorial," *Foundations and Trends® in Computer Graphics and Vision*, vol. 9, no. 1-2, pp. 1–148, 2015. 48

[202] S. Li, "Binocular spherical stereo," *IEEE Transactions on intelligent transportation systems*, vol. 9, no. 4, pp. 589–600, 2008. 48

[203] J. Huang, Z. Chen, D. Ceylan, and H. Jin, "6-DOF VR videos with a single 360-camera," in *2017 IEEE Virtual Reality (VR)*. IEEE, 2017, pp. 37–44. 48

[204] R. Khasanova and P. Frossard, "Graph-based classification of omnidirectional images," in *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 869–878. 49

[205] Y.-C. Su and K. Grauman, "Learning Spherical Convolution for Fast Features from 360° Imagery." in *NIPS*, vol. 2, no. 3, 2017, p. 5. 49

[206] Y. Jeon and J. Kim, "Active convolution: Learning the shape of convolution for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4201–4209. 49

[207] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 764–773. 49, 50, 51, 80, 81

[208] L. Deng, M. Yang, H. Li, T. Li, B. Hu, and C. Wang, "Restricted deformable convolution-based road scene semantic segmentation using surround view cameras," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 10, pp. 4350–4362, 2019. 49, 55

[209] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587. 49, 50, 51

[210] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6154–6162. 50, 51

[211] R. Girshick, F. Iandola, T. Darrell, and J. Malik, "Deformable part models are convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 437–446. 50, 51

[212] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 734–750. 50, 52

[213] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Craft objects from images," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 6043–6051. 50, 51

[214] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy *et al.*, "Deepid-net: Deformable deep convolutional neural networks for object detection," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2403–2412. 50, 51

[215] X. Zhou, J. Zhuo, and P. Krahenbuhl, "Bottom-up object detection by grouping extreme and center points," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 850–859. 50, 52

[216] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2980–2988. 50, 52, 143, 174

[217] W. Ouyang, K. Wang, X. Zhu, and X. Wang, "Chained cascade network for object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1938–1946. 50, 51

[218] T. Mordan, N. Thome, M. Cord, and G. Henaff, "Deformable part-based fully convolutional network for object detection," in *British Machine Vision Conference 2017, BMVC*. BMVA Press, 2017. 50, 51

[219] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6569–6578. 50, 52

[220] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 129–137. 50, 51

[221] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, 2016, pp. 379–387. 50

[222] S. Zagoruyko, A. Lerer, T.-Y. Lin, P. O. Pinheiro, S. Gross, S. Chintala, and P. Dollár, "A multipath network for object detection," in *Proceedings of the British Machine Vision Conference 2016, BMVC*. BMVA Press, 2016. 50, 51

[223] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015. 49, 50, 54, 55

[224] S. Gidaris and N. Komodakis, "Object detection via a multi-region and semantic segmentation-aware cnn model," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1134–1142. 50

[225] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional

networks," in *2nd International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2014. 50

[226] T. Kong, A. Yao, Y. Chen, and F. Sun, "Hypernet: Towards accurate region proposal generation and joint object detection," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 845–853. 50, 51

[227] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD: Deconvolutional single shot detector," *arXiv preprint arXiv:1701.06659*, 2017. 50, 52

[228] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang, "Acquisition of localization confidence for accurate object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 784–799. 50, 51

[229] L. Cui, "MDSSD: multi-scale deconvolutional single shot detector for small objects," *Science China Information Sciences*, vol. 63, 2020. 50

[230] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang *et al.*, "Hybrid task cascade for instance segmentation," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4974–4983. 50

[231] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 213–229. 50, 51

[232] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 781–10 790. 50, 51

[233] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271. 50

[234] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018. 50, 129, 133, 142

[235] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 6105–6114. 51

[236] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, 2013, pp. 2056–2063. 51

[237] M. Lourenço, J. P. Barreto, and F. Vasconcelos, "sRD-SIFT: Keypoint detection and matching in images with radial distortion," *IEEE Transactions on Robotics*, vol. 28, no. 3, pp. 752–760, 2012. 52

[238] S. Agarwal, J. O. D. Terrail, and F. Jurie, "Recent advances in object detection in the age of deep convolutional neural networks," *arXiv preprint arXiv:1809.03193*, 2018. 52

[239] T. Li, G. Tong, H. Tang, B. Li, and B. Chen, "Fisheyedet: A self-study and contour-based object detector in fisheye images," *IEEE Access*, vol. 8, pp. 71 739–71 751, 2020. 52

[240] B. Coors, A. P. Condurache, and A. Geiger, "Spherenet: Learning spherical representations for detection and classification in omnidirectional images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 518–533. 52

[241] N. Perraudin, M. Defferrard, T. Kacprzak, and R. Sgier, "DeepSphere: Efficient spherical convolutional neural network with HEALPix sampling for cosmological applications," *Astronomy and Computing*, vol. 27, pp. 130–146, 2019. 52

[242] Y.-C. Su and K. Grauman, "Kernel transformer networks for compact spherical convolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9442–9451. 52, 172

[243] C. Jiang, J. Huang, K. Kashinath, P. Marcus, M. Niessner *et al.*, "Spherical CNNs on unstructured grids," in *7th International Conference on Learning Representations, ICLR, 2019*. OpenReview.net, 2019. 52, 56

[244] F. Deng, X. Zhu, and J. Ren, "Object detection on panoramic images based on deep learning," in *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, 2017, pp. 375–380. 52

[245] W. Yang, Y. Qian, J.-K. Kämäräinen, F. Cricri, and L. Fan, "Object detection in equirectangular panorama," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 2190–2195. 52

[246] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 35, no. 8, pp. 1915–1929, 2012. 53

[247] P. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *International conference on machine learning*. PMLR, 2014, pp. 82–90. 53

[248] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014. 53

[249] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241. 53, 55, 79

[250] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR*, 2015. 53, 55

[251] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *4th International Conference on Learning Representations, ICLR*, 2016. 53

[252] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1529–1537. 53

[253] G. Lin, C. Shen, A. Van Den Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3194–3203. 54

[254] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, "Semantic object parsing with graph lstm," in *European Conference on Computer Vision*. Springer, 2016, pp. 125–143. 54

[255] B. Shuai, Z. Zuo, B. Wang, and G. Wang, "Dag-recurrent neural networks for scene labeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3620–3629. 54

[256] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017. 54, 55

[257] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890. 54, 55

[258] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017. 54, 55

[259] P. Luo, G. Wang, L. Lin, and X. Wang, "Deep dual learning for semantic image segmentation," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2718–2726. 54

[260] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters–improve semantic segmentation by global convolutional network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4353–4361. 54

[261] J. Fu, J. Liu, Y. Wang, J. Zhou, C. Wang, and H. Lu, "Stacked deconvolutional network for semantic segmentation," *IEEE Transactions on Image Processing*, 2019. 54

[262] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Learning a discriminative feature network for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1857–1866. 54

[263] D. Lin, Y. Ji, D. Lischinski, D. Cohen-Or, and H. Huang, "Multi-scale context intertwining for semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 603–619. 54

[264] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 801–818. 54, 55

[265] H. Shi, H. Li, F. Meng, Q. Wu, L. Xu, and K. N. Ngan, "Hierarchical parsing net: Semantic scene parsing from global scene to objects," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2670–2682, 2018. 54

[266] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, "Context encoding for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7151–7160. 54

[267] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. C. Loy, D. Lin, and J. Jia, "Psanet: Point-wise spatial attention network for scene parsing," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 267–283. 54

[268] X. Li, Z. Zhong, J. Wu, Y. Yang, Z. Lin, and H. Liu, "Expectation-maximization attention networks for semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9167–9176. 54

[269] Y. Huang, Q. Wang, W. Jia, and X. He, "See More Than Once–Kernel-Sharing Atrous Convolution for Semantic Segmentation," *arXiv preprint arXiv:1908.09443*, 2019. 54

[270] H. Zhang, H. Zhang, C. Wang, and J. Xie, "Co-occurrent features in semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 548–557. 55

[271] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708. 55

[272] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1251–1258. 55

[273] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012. 55

[274] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 82–92. 55

[275] L.-C. Chen, M. Collins, and Y. Zhu, "George, Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens," *Searching for efficient multi-scale, architectures for dense image prediction. In, NeurIPS*, vol. 1, no. 2, 2018. 55

[276] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016. 55

[277] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 325–341. 55

[278] G. Li, I. Yun, J. Kim, and J. Kim, "Dabnet: Depth-wise asymmetric bottleneck for real-time semantic segmentation," in *30th British Machine Vision Conference, BMVC 2019*. BMVA Press, 2019, p. 259. 55

[279] H. Li, P. Xiong, H. Fan, and J. Sun, "Dfanet: Deep feature aggregation for real-time semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9522–9531. 55

[280] P. Chao, C.-Y. Kao, Y.-S. Ruan, C.-H. Huang, and Y.-L. Lin, "Hardnet: A low memory traffic network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3552–3561. 55

[281] L. Deng, M. Yang, Y. Qian, C. Wang, and B. Wang, "CNN based semantic segmentation for urban traffic scenes using fisheye camera," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 231–236. 55

[282] Á. Sáez, L. M. Bergasa, E. López-Guillén, E. Romera, M. Tradacete, C. Gómez-Huélamo, and J. Del Egido, "Real-time semantic segmentation for fisheye urban driving images based on ERFNet," *Sensors*, vol. 19, no. 3, p. 503, 2019. 55

[283] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2017. 55

[284] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223. 55

[285] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 3234–3243. 55

[286] Y. Xu, K. Wang, K. Yang, D. Sun, and J. Fu, "Semantic segmentation of panoramic images using a synthetic dataset," in *Artificial Intelligence and Machine Learning in Defense Applications*, vol. 11169. International Society for Optics and Photonics, 2019, p. 111690B. 55

[287] K. Yang, X. Hu, L. M. Bergasa, E. Romera, X. Huang, D. Sun, and K. Wang, "Can we pass beyond the field of view? panoramic annular semantic segmentation for real-world surrounding perception," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 446–453. 55

[288] R. Monroy, S. Lutz, T. Chalasani, and A. Smolic, "Salnet360: Saliency maps for omni-directional images with cnn," *Signal Processing: Image Communication*, vol. 69, pp. 26–34, 2018. 56

[289] W.-S. Lai, Y. Huang, N. Joshi, C. Buehler, M.-H. Yang, and S. B. Kang, "Semantic-driven generation of hyperlapse from 360 degree video," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 9, pp. 2610–2621, 2017. 56

[290] K. Tateno, N. Navab, and F. Tombari, "Distortion-aware convolutional filters for dense prediction in panoramic images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 707–722. 56

[291] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, "Spherical cnns," in *6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*. OpenReview.net, 2018. 56

[292] Y. Lee, J. Jeong, J. Yun, W. Cho, and K.-J. Yoon, "Spherephd: Applying cnns on a spherical polyhedron representation of 360deg images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9181–9189. 56

[293] C. Zhang, S. Liwicki, W. Smith, and R. Cipolla, "Orientation-aware semantic segmentation on icosahedron spheres," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, 2019, pp. 3533–3541. 56

[294] S. Wehrwein and R. Szeliski, "Video segmentation with background motion models." in *BMVC*, vol. 245, 2017, p. 246. 56

[295] H. Rashed, M. Ramzy, V. Vaquero, A. El Sallab, G. Sistu, and S. Yogamani, "Fusemodnet: Real-time camera and lidar based moving object detection for robust low-light autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. IEEE, 2019, pp. 2393–2402. 57

[296] S. D. Jain, B. Xiong, and K. Grauman, "Fusionseg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos," in *2017 IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 2117–2126. 57

[297] B. Drayer and T. Brox, "Object detection, tracking, and motion segmentation for object-level video segmentation," *arXiv preprint arXiv:1608.03066*, 2016. 57

[298] S. Dey, V. Reilly, I. Saleemi, and M. Shah, "Detection of independently moving objects in non-planar scenes via multi-frame monocular epipolar constraint," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2012, pp. 860–873. 57

[299] J. C. Clarke and A. Zisserman, "Detection and tracking of independent motion," *Image and Vision Computing*, vol. 14, no. 8, pp. 565–572, 1996. 57

[300] J. Klappstein, F. Stein, and U. Franke, "Monocular motion detection using spatial constraints in a unified manner," in *2006 IEEE Intelligent Vehicles Symposium*, 2006, pp. 261–267. 57

[301] P. Spagnolo, M. Leo, A. Distante *et al.*, "Moving object segmentation by background subtraction and temporal analysis," *Image and Vision Computing*, vol. 24, no. 5, pp. 411–423, 2006. 57

[302] P. Gao, X. Sun, and W. Wang, "Moving object detection based on kirsch operator combined with Optical Flow," in *2010 International Conference on Image Analysis and Signal Processing*. IEEE, 2010, pp. 620–624. 57

[303] A. Ranjan, V. Jampani, L. Balles, K. Kim, D. Sun, J. Wulff, and M. J. Black, "Competitive Collaboration: Joint Unsupervised Learning of Depth, Camera Motion, Optical Flow and Motion Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 240–12 249. 57, 62, 82, 90, 110, 112, 123, 153

[304] F. Tosi, F. Aleotti, P. Z. Ramirez, M. Poggi, S. Salti, L. D. Stefano, and S. Mattoccia, "Distilled semantics for comprehensive scene understanding from videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4654–4665. 57, 112, 181

[305] W. Choi, C. Pantofaru, and S. Savarese, "A general framework for tracking multiple people from a moving camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 35, no. 7, pp. 1577–1591, 2012. 57

[306] T. Chen and S. Lu, "Object-level motion detection from moving cameras," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 11. IEEE, 2016, pp. 2333–2343. 57

[307] N. D. Reddy, P. Singhal, and K. M. Krishna, "Semantic motion segmentation using dense CRF formulation," in *Proceedings of the 2014 Indian conference on computer vision graphics and image processing*, 2014, pp. 1–8. 57

[308] Q. Fan, Y. Yi, L. Hao, F. Mengyin, and W. Shunting, "Semantic motion segmentation for urban dynamic scene understanding," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2016, pp. 497–502. 57

[309] H. Wang, P. Wang, and X. Qian, "MPNET: An end-to-end deep neural network for object detection in surveillance video," *IEEE Access*, vol. 6, pp. 30 296–30 308, 2018. 57

[310] M. Siam, S. Eikerdawy, M. Gamal, M. Abdel-Razek, M. Jagersand, and H. Zhang, "Real-time segmentation with appearance, motion and geometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5793–5800. 57

[311] H. Porav, T. Bruls, and P. Newman, "I can see clearly now: Image restoration via de-raining," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7087–7093. 58

[312] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic foggy scene understanding with synthetic data," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 973–992, 2018. 58

[313] S. Li, I. B. Araujo, W. Ren, Z. Wang, E. K. Tokuda, R. H. Junior, R. Cesar-Junior, J. Zhang, X. Guo, and X. Cao, "Single image deraining: A comprehensive benchmark analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3838–3847. 58

[314] D. Ren, W. Zuo, Q. Hu, P. Zhu, and D. Meng, "Progressive image deraining networks: A better and simpler baseline," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3937–3946. 58

[315] T. Wang, X. Yang, K. Xu, S. Chen, Q. Zhang, and R. W. Lau, "Spatial attentive single-image deraining with a high quality real rain dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 270–12 279. 58

[316] W. Yang, J. Liu, and J. Feng, "Frame-consistent recurrent video deraining with dual-level flow," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1661–1670. 58

[317] W. Yang, J. Liu, S. Yang, and Z. Guo, "Scale-free single image deraining via visibility-enhanced recurrent wavelet learning," *IEEE Transactions on Image Processing*, vol. 28, no. 6, pp. 2948–2961, 2019. 58

[318] A. Das, "Soildnet: Soiling degradation detection in autonomous driving," *arXiv preprint arXiv:1911.01054*, 2019. 58

[319] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, "Multi-Task Learning for Dense Prediction Tasks: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 59, 60, 63, 139

[320] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 109–117. 59

[321] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram, "Multi-Task Learning for Classification with Dirichlet Process Priors." *Journal of Machine Learning Research*, vol. 8, no. 1, 2007. 59

[322] L. Jacob, F. Bach, and J.-P. Vert, "Clustered Multi-Task Learning: A Convex Formulation," in *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*. Curran Associates, Inc., 2008, pp. 745–752. 59

[323] J. Zhou, J. Chen, and J. Ye, "Clustered multi-task learning via alternating structure optimization," *Advances in neural information processing systems*, vol. 2011, p. 702, 2011. 59

[324] B. Bakker and T. Heskes, "Task clustering and gating for bayesian multitask learning," *J. Mach. Learn. Res.*, vol. 4, pp. 83–99, 2003. 59

[325] K. Yu, V. Tresp, and A. Schwaighofer, "Learning Gaussian processes from multiple tasks," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 1012–1019. 59

[326] S.-I. Lee, V. Chatalbashev, D. Vickrey, and D. Koller, "Learning a meta-level prior for feature relevance from multiple related tasks," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 489–496. 59

[327] H. Daumé III, "Bayesian multitask learning with latent hierarchies," in *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*. AUAI Press, 2009, pp. 135–142. 59

[328] A. Kumar and H. Daume III, "Learning task grouping and overlap in multi-task learning," in *Proceedings of the 29th International Conference on Machine Learning, ICML*. icml.cc / Omnipress, 2012. 59

[329] A. Argyriou, T. Evgeniou, and M. Pontil, "Convex multi-task feature learning," *Machine learning*, vol. 73, no. 3, pp. 243–272, 2008. 59

[330] J. Liu, S. Ji, and J. Ye, "Multi-task feature learning via efficient l2, 1-norm minimization," in *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*. AUAI Press, 2009, pp. 339–348. 59

[331] A. Jalali, S. Sanghavi, C. Ruan, and P. Ravikumar, "A dirty model for multi-task learning," *Advances in neural information processing systems*, vol. 23, pp. 964–972, 2010. 59

[332] A. Agarwal, S. Gerber, and H. Daume, "Learning multiple tasks using manifold regularization," in *Advances in neural information processing systems*, 2010, pp. 46–54. 59

[333] R. K. Ando, T. Zhang, and P. Bartlett, "A framework for learning predictive structures from multiple tasks and unlabeled data." *Journal of Machine Learning Research*, vol. 6, no. 11, 2005. 59

[334] P. Rai and H. Daumé III, "Infinite predictor subspace models for multitask learning," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 613–620. 59

[335] D. Xu, W. Ouyang, X. Wang, and N. Sebe, "Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 675–684. 59, 61

[336] Z. Zhang, Z. Cui, C. Xu, Z. Jie, X. Li, and J. Yang, "Joint task-recursive learning for semantic segmentation and depth estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 235–251. 59, 61

[337] Z. Zhang, Z. Cui, C. Xu, Y. Yan, N. Sebe, and J. Yang, "Pattern-affinitive propagation across depth, surface normal and semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4106–4115. 59, 61

[338] L. Zhou, Z. Cui, C. Xu, Z. Zhang, C. Wang, T. Zhang, and J. Yang, "Pattern-structure diffusion for multi-task learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4514–4523. 59

[339] S. Vandenhende, S. Georgoulis, and L. Van Gool, "Mti-net: Multi-scale task interaction networks for multi-task learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 527–543. 59, 61

[340] M. Crawshaw, "Multi-Task Learning with Deep Neural Networks: A Survey," *arXiv preprint arXiv:2009.09796*, 2020. 60

[341] Y. Gao, J. Ma, M. Zhao, W. Liu, and A. L. Yuille, "Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3205–3214. 60, 61

[342] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1871–1880. 61, 63, 148, 149

[343] M. Long, Z. Cao, J. Wang, and P. S. Yu, "Learning multiple tasks with multilinear relationship networks," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 2017, pp. 1594–1603. 61

[344] E. Meyerson and R. Miikkulainen, "Beyond Shared Hierarchies: Deep Multitask Learning through Soft Layer Ordering," in *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net, 2018. 61

[345] Y. Yang and T. M. Hospedales, "Deep Multi-task Representation Learning: A Tensor Factorisation Approach," in *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net, 2017. 61

[346] C. Rosenbaum, T. Klinger, and M. Riemer, "Routing networks: Adaptive selection of non-linear functions for multi-task learning," in *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net, 2018. 61

[347] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 67–82. 61

[348] S. Huang, X. Li, Z.-Q. Cheng, Z. Zhang, and A. Hauptmann, "Gnas: A greedy neural architecture search method for multi-attribute learning," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 2049–2057. 61

[349] F. J. Bragman, R. Tanno, S. Ourselin, D. C. Alexander, and J. Cardoso, "Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1385–1394. 61

[350] A. Newell, L. Jiang, C. Wang, L.-J. Li, and J. Deng, "Feature partitioning for efficient multi-task architectures," *arXiv preprint arXiv:1908.04339*, 2019. 61

[351] K.-K. Maninis, I. Radosavovic, and I. Kokkinos, "Attentive single-tasking of multiple tasks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1851–1860. 61

[352] C. Mao, A. Gupta, V. Nitin, B. Ray, S. Song, J. Yang, and C. Vondrick, "Multitask Learning Strengthens Adversarial Robustness," in *Computer Vision – ECCV 2020*. Springer International Publishing, 2020, pp. 158–174. 62, 66, 139

[353] M. Klingner, A. Bar, and T. Fingscheidt, "Improved Noise and Attack Robustness for Semantic Segmentation by Using Multi-Task Training with Self-Supervised Depth Estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020, pp. 320–321. 62

[354] M. Klingner, J.-A. Termöhlen, J. Mikolajczyk, and T. Fingscheidt, "Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 582–600. 62, 112

[355] S. Chennupati, G. Sistu, S. Yogamani, and S. A Rawashdeh, "Multinet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. Computer Vision Foundation / IEEE, 2019, pp. 1200–1210. 62, 148, 149

[356] J.-A. Bolte, M. Kamp, A. Breuer, S. Homoceanu, P. Schlicht, F. Huger, D. Lipinski, and T. Fingscheidt, "Unsupervised domain adaptation to improve image segmentation quality both in the source and target domain," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019*, 2019, pp. 1404–1413. 62

[357] M. Ochs, A. Kretz, and R. Mester, "SDNet: Semantically guided depth estimation network," in *German Conference on Pattern Recognition (GCPR)*. Springer, 2019, pp. 288–302. 62

[358] S. Zhao, H. Fu, M. Gong, and D. Tao, "Geometry-aware symmetric domain adaptation for monocular depth estimation," in *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 9788–9798. 62

[359] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6399–6408. 62

[360] L. Wang, J. Zhang, O. Wang, Z. Lin, and H. Lu, "SDC-depth: Semantic divide-and-conquer network for monocular depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 541–550. 62

[361] Y. Lu, M. Sarkis, and G. Lu, "Multi-Task Learning for Single Image Depth Estimation and Segmentation Based on Unsupervised Network," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10 788–10 794. 62

[362] Y. Meng, Y. Lu, A. Raj, S. Sunarjo, R. Guo, T. Javidi, G. Bansal, and D. Bharadia, "Signet: Semantic instance aided unsupervised 3d geometry perception," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9810–9820. 62, 112

[363] C. Luo, Z. Yang, P. Wang, Y. Wang, W. Xu, R. Nevatia, and A. Yuille, "Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2624–2641, 2019. 62, 76, 82, 90, 110, 111, 123, 153, 169

[364] Y. Chen, C. Schmid, and C. Sminchisescu, "Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7063–7072. 62, 109, 112, 152

[365] P. Liu, M. Lyu, I. King, and J. Xu, "Selflow: Self-supervised learning of optical flow," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4571–4580. 62

[366] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha, "Unsupervised deep learning for optical flow estimation," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 1495–1501. 62

[367] L. Liu, G. Zhai, W. Ye, and Y. Liu, "Unsupervised Learning of Scene Flow Estimation Fusing with Local Rigidity." in *Ijcai*, 2019, pp. 876–882. 62

[368] Y. Wang, P. Wang, Z. Yang, C. Luo, Y. Yang, and W. Xu, "Unos: Unified unsupervised optical-flow and stereo-depth estimation by watching videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8071–8081. 62

[369] Z. Yang, P. Wang, Y. Wang, W. Xu, and R. Nevatia, "Every pixel counts: Unsupervised geometry learning with holistic 3d motion understanding," in *Proceedings of the European Conference on Computer Vision (ECCV) 2018 Workshops*, vol. 11133. Springer, 2018, pp. 691–709. 62, 112

[370] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," in *Proceedings of the IEEE Conference on Computer Vision and*

Pattern Recognition (CVPR), 2018, pp. 1983–1992. 62, 77, 80, 82, 84, 110, 112, 123, 153

[371] P.-Y. Chen, A. H. Liu, Y.-C. Liu, and Y.-C. F. Wang, "Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2624–2632. 62, 63

[372] J. Novosel, P. Viswanath, and B. Arsenali, "Boosting semantic segmentation with multi-task self-supervised learning for autonomous driving applications," in *Proceedings of NeurIPS-Workshops*, vol. 3, 2019. 62

[373] G. Yang, H. Zhao, J. Shi, Z. Deng, and J. Jia, "Segstereo: Exploiting semantic information for disparity estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 636–651. 62, 63

[374] S. Zhu, G. Brazil, and X. Liu, "The edge of depth: Explicit constraints between segmentation and depth," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 13 116–13 125. 62, 63

[375] M. Guo, A. Haque, D.-A. Huang, S. Yeung, and L. Fei-Fei, "Dynamic task prioritization for multitask learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 270–287. 63, 148, 149

[376] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*. Pmlr, 2015, pp. 1180–1189. 63

[377] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR*, 2014. 64, 65

[378] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015*. OpenReview.net, 2015. 64, 65

[379] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *International conference on machine learning*. Pmlr, 2018, pp. 284–293. 64

[380] A. Kurakin, I. Goodfellow, S. Bengio *et al.*, "Adversarial examples in the physical world," in *5th International Conference on Learning Representations, ICLR, Workshop Track Proceedings*. OpenReview.net, 2017. 64, 65

[381] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016, pp. 372–387. 64, 65

[382] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019. 64

[383] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks. arXiv e-prints, page," in *IEEE Symposium on Security and Privacy, SP*. IEEE Computer Society, 2016, pp. 39–57. 64, 65

[384] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 582–597. 64

[385] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2574–2582. 64, 65

[386] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1765–1773. 64, 65

[387] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *Ieee Access*, vol. 6, pp. 14 410–14 430, 2018. 64, 65

[388] S. Sarkar, A. Bansal, U. Mahbub, and R. Chellappa, "UPSET and ANGRI: Breaking high performance image classifiers," *arXiv preprint arXiv:1707.01159*, 2017. 64, 65

[389] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, "Houdini: Fooling deep structured prediction models," *arXiv preprint arXiv:1707.05373*, 2017. 64

[390] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*. PMLR, 2016, pp. 173–182. 64

[391] S. Baluja and I. Fischer, "Adversarial transformation networks: Learning to generate adversarial examples," *arXiv preprint arXiv:1703.09387*, 2017. 64

[392] J. Hayes and G. Danezis, "Machine learning as an adversarial service: Learning black-box adversarial examples," *arXiv preprint arXiv:1708.05207*, vol. 2, 2017. 64

[393] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," in *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2017, pp. 446–454. 64

[394] A. Arnab, O. Miksik, and P. H. Torr, "On the robustness of semantic segmentation models to adversarial attacks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 888–897. 65

[395] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2017, pp. 1369–1378. 65

[396] T. v. Dijk and G. d. Croon, "How do neural networks see depth in single images?" in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 2183–2191. 65, 166, 167

[397] K. Yamanaka, R. Matsumoto, K. Takahashi, and T. Fujii, "Adversarial Patch Attacks on Monocular Depth Estimation Networks," *IEEE Access*, vol. 8, pp. 179 094–179 104, 2020. 65

[398] K. R. Mopuri, U. Garg, and R. V. Babu, "Fast feature fool: A data independent approach to universal adversarial perturbations," in *British Machine Vision Conference BMVC*. BMVA Press, 2017. 65

[399] J. Hu and T. Okatani, "Analysis of deep networks for monocular depth estimation through adversarial attacks with proposal of a defense method," *arXiv preprint arXiv:1911.08790*, 2019. 65

[400] S. Thys, W. Van Ranst, and T. Goedemé, "Fooling automated surveillance cameras: adversarial patches to attack person detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019, pp. 49–55. 65

[401] Y. Cao, C. Xiao, D. Yang, J. Fang, R. Yang, M. Liu, and B. Li, "Adversarial objects against lidar-based autonomous driving systems," *arXiv preprint arXiv:1907.05418*, 2019. 66

[402] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1625–1634. 66

[403] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of jpg compression on adversarial images," *arXiv preprint arXiv:1608.00853*, 2016. 66, 154

[404] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *25th Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2018. 66

[405] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models," in *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net, 2018. 66

[406] C. Pinard, "Robust Learning of a depth map for obstacle avoidance with a monocular stabilized flying camera. (Apprentissage robuste d'une carte de profondeur pour l'évitement d'obstacle dans le cas des cameras volantes, monoculaires et stabilisées)," Ph.D. dissertation, University of Paris-Saclay, France, 2019. [Online]. Available: https://tel.archives-ouvertes.fr/tel-02285215 69, 166, 167, 172

[407] "Computer Science Tripos Part II," May 2021. [Online]. Available: https://www.cl.cam.ac.uk/teaching/1011/CompVision/Town 69

[408] Z. Arican and P. Frossard, "Dense disparity estimation from omnidirectional images," in *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*. IEEE, 2007, pp. 399–404. 70

[409] "Lecture on SfM approaches," May 2021. [Online]. Available: https://www.cs.unc.edu/~lazebnik/spring11/lec17%5Fsfm.pdf 70

[410] P. Yadati and A. M. Namboodiri, "Multiscale two-view stereo using convolutional neural networks for unrectified images," in *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*. IEEE, 2017, pp. 346–349. 71

[411] E. Meijering, “A chronology of interpolation: from ancient astronomy to modern signal and image processing,” *Proceedings of the IEEE*, vol. 90, no. 3, pp. 319–342, 2002. 71

[412] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, “Spatial transformer networks,” in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, 2015, pp. 2017–2025. 75, 77, 78

[413] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, “Loss functions for image restoration with neural networks,” *IEEE Transactions on computational imaging*, vol. 3, no. 1, pp. 47–57, 2016. 75

[414] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004. 75, 141, 171

[415] L. Zhou, J. Ye, M. Abello, S. Wang, and M. Kaess, “Unsupervised learning of monocular depth estimation with bundle adjustment, super-resolution and clip loss,” *arXiv preprint arXiv:1812.03368*, 2018. 76, 77, 82, 165

[416] Y. Zou, Z. Luo, and J.-B. Huang, “Df-net: Unsupervised joint learning of depth and flow using cross-task consistency,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 36–53. 76, 77, 82, 110, 112, 153

[417] X. Zhu, H. Hu, S. Lin, and J. Dai, “Deformable convnets v2: More deformable, better results,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9308–9316. 80, 84, 165

[418] J. Li, Y. Chen, L. Cai, I. Davidson, and S. Ji, “Dense transformer networks,” *arXiv preprint arXiv:1705.08881*, 2017. 80

[419] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1874–1883. 80, 81, 165

[420] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, “On the variance of the adaptive learning rate and beyond,” in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 2020*. OpenReview.net, 2020. 81, 149

[421] M. Zhang, J. Lucas, J. Ba, and G. E. Hinton, “Lookahead Optimizer: k steps forward, 1 step back,” in *Advances in Neural Information Processing Systems*, 2019, pp. 9593–9604. 81, 149

[422] L. N. Smith and N. Topin, “Super-convergence: Very fast training of neural networks using large learning rates,” in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. International Society for Optics and Photonics, 2019. 81

[423] A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi, “Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize,” *arXiv preprint arXiv:1707.02937*, 2017. 81, 83, 84

[424] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning*. PMLR, 2015, pp. 448–456. 81

[425] Y. Wu and K. He, “Group normalization,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19. 81

[426] K. He, R. Girshick, and P. Dollár, “Rethinking imagenet pre-training,” in *Proceedings of the IEEE international conference on computer vision*, 2019, pp. 4918–4927. 81

[427] Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia, “Unsupervised learning of geometry from videos with edge-aware depth-normal consistency,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018. 82, 112

[428] A. Odena, V. Dumoulin, and C. Olah, “Deconvolution and checkerboard artifacts. Distill (2016),” 2016. 83, 84, 165

[429] H. Hu, Z. Zhang, Z. Xie, and S. Lin, “Local relation networks for image recognition,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3464–3473. 96, 173

[430] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, “Stand-alone self-attention in vision models,” in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS) 2019*, 2019, pp. 68–80. 96, 97, 172, 173

[431] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proceedings of NIPS*, Dec. 2017, pp. 5998–6008. 97, 172, 173

[432] J. M. Facil, B. Ummenhofer, H. Zhou, L. Montesano, T. Brox, and J. Civera, “CAM-Convs: camera-aware multi-scale convolutions for single-view depth,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 11 826–11 835. 101, 103

[433] J. P. Barreto, “Unifying image plane liftings for central catadioptric and dioptric cameras,” in *Imaging Beyond the Pinhole Camera*. Springer, 2006, pp. 21–38. 102, 173

[434] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski, “An Intriguing Failing of Convolutional Neural Networks and the Coordconv solution,” in *Proceedings of NIPS*, Montréal, QC, Canada, Dec. 2018, pp. 9605–9616. 103

[435] H. Su, V. Jampani, D. Sun, O. Gallo, E. Learned-Miller, and J. Kautz, “Pixel-adaptive convolutional neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 11 166–11 175. 103

[436] M. Klodt and A. Vedaldi, “Supervising the new with the old: learning SFM from SFM,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 698–713. 110, 153

[437] Z. Yang, P. Wang, Y. Wang, W. Xu, and R. Nevatia, "Lego: Learning edge with geometry all at once by watching videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 225–234. 112

[438] G. Wang, H. Wang, Y. Liu, and W. Chen, "Unsupervised learning of monocular depth and ego-motion using multiple masks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4724–4730. 112

[439] Q. Sun, Y. Tang, and C. Zhao, "Cycle-SfM: Joint self-supervised learning of depth and camera motion from monocular image sequences," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, no. 12, p. 123102, 2019. 112

[440] S. Li, F. Xue, X. Wang, Z. Yan, and H. Zha, "Sequential adversarial learning for self-supervised deep visual odometry," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2851–2860. 112

[441] Y. Almalioglu, M. R. U. Saputra, P. P. de Gusmao, A. Markham, and N. Trigoni, "GANVO: Unsupervised deep monocular visual odometry and depth estimation with generative adversarial networks," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 5474–5480. 112

[442] J.-W. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid, "Unsupervised scale-consistent depth and ego-motion learning from monocular video," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS) 2019*, 2019, pp. 35–45. 112

[443] S. Pillai, R. Ambruş, and A. Gaidon, "Superdepth: Self-supervised, super-resolved monocular depth estimation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9250–9256. 112

[444] V. Patil, W. Van Gansbeke, D. Dai, and L. Van Gool, "Don't forget the past: Recurrent depth estimation from monocular video," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6813–6820, 2020. 112, 180

[445] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 340–349. 123

[446] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Proceedings. international conference on image processing*, vol. 1, 2002, pp. I–i. 125

[447] W. Dong, P. Roy, C. Peng, and V. Isler, "Ellipse r-cnn: Learning to infer elliptical object from clustering and occlusion," *IEEE Transactions on Image Processing*, vol. 30, pp. 2193–2206, 2021. 126

[448] C. Brauer-Burchardt and K. Voss, "A new algorithm to correct fish-eye-and strong wide-angle-lens-distortion from single images," in *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, vol. 1. IEEE, 2001, pp. 225–228. 126

[449] C. Hughes, R. McFeely, P. Denny, M. Glavin, and E. Jones, "Equidistant fish-eye perspective with application in distortion centre estimation," *Image and Vision Computing*, vol. 28, pp. 538–551, 2010. 126, 127

[450] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A flexible technique for accurate omnidirectional camera calibration and structure from motion," in *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*. IEEE, 2006, pp. 45–45. 127

[451] E. Xie, P. Sun, X. Song, W. Wang, X. Liu, D. Liang, C. Shen, and P. Luo, "Polarmask: Single shot instance segmentation with polar representation," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 12 193–12 202. 128

[452] P. Hurtik, V. Molek, J. Hula, M. Vajgl, P. Vlasanek, and T. Nejezchleba, "Poly-YOLO: higher speed, more precise detection and instance segmentation for YOLOv3," *arXiv preprint arXiv:2005.13243*, 2020. 128, 129, 131

[453] C.-H. Teh and R. T. Chin, "On the detection of dominant points on digital curves," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 11, no. 8, pp. 859–872, 1989. 129

[454] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: the international journal for geographic information and geovisualization*, vol. 10, no. 2, pp. 112–122, 1973. 129

[455] P. Maddu, W. Doherty, G. Sistu, I. Leang, M. Uricar, S. Chennupati, H. Rashed, J. Horgan, C. Hughes, and S. Yogamani, "FisheyeMultiNet: Real-time Multi-task Learning Architecture for Surround-view Automated Parking System," in *Irish Machine Vision and Image Processing Conference*, 2019. 139

[456] M. Berman, A. R. Triki, and M. B. Blaschko, "The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4413–4421. 142, 143

[457] A. Saxena, S. H. Chung, and A. Y. Ng, "3-d depth reconstruction from a single still image," *International journal of computer vision*, vol. 76, no. 1, pp. 53–69, 2008. 166

[458] I. C. McManus, J. Buckman, and E. Woolley, "Is light in pictures presumed to come from the left side?" *Perception*, vol. 33, no. 12, pp. 1421–1436, 2004. 166

[459] "Ames Room Photograph," May 2021. [Online]. Available: https://en.wikipedia.org/wiki/File:Ames%5Froom%5Fforced%5Fperspective.jpg 166

[460] "Dakar 2019 Photograph," May 2021. [Online]. Available: https://www.straitstimes.com/multimedia/photos/leaving-his-rival-bikers-in-the-dust 166

[461] D. M. Eagleman, "Visual illusions and neurobiology," *Nature Reviews Neuroscience*, vol. 2, no. 12, pp. 920–926, 2001. 167

[462] "SFM Self Supervised Depth Estimation: Breaking Down The Ideas," Apr. 2021. [Online]. Available: https://towardsdatascience.com/self-supervised-depth-estimation-breaking-down-the-ideas-f212e4f05ffa 169, 170, 172

[463] X. Weng and K. Kitani, "Monocular 3d object detection with pseudo-lidar point cloud," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 857–866. 170, 171

[464] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 33, no. 3, pp. 500–513, 2010. 172

[465] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989. 173

[466] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7794–7803. 173

[467] "YOLOv3: Real-Time Object Detection Algorithm (What's New?)," May 2021. [Online]. Available: https://viso.ai/deep-learning/yolov3-overview/ 174

[468] Y. S. Abu-Mostafa, "Learning from hints in neural networks," *Journal of complexity*, vol. 6, no. 2, pp. 192–198, 1990. 175

[469] S. Liu, Y. Liang, and A. Gitter, "Loss-balanced task weighting to reduce negative transfer in multi-task learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 9977–9978. 175, 176

[470] S. Wu, H. R. Zhang, and C. Ré, "Understanding and improving information transfer in multi-task learning," in *8th International Conference on Learning Representations, ICLR*. OpenReview.net, 2020. 176, 177

[471] K. Karsch, C. Liu, and S. B. Kang, "Depth transfer: Depth extraction from video using non-parametric sampling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 11, pp. 2144–2158, 2014. 180

[472] Z. Li, X. Liu, F. X. Creighton, R. H. Taylor, and M. Unberath, "Revisiting Stereo Depth Estimation From a Sequence-to-Sequence Perspective with Transformers," *arXiv preprint arXiv:2011.02910*, 2020. 180

[473] J. Hu, Y. Zhang, and T. Okatani, "Visualization of convolutional neural networks for monocular depth estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3869–3878. 180

[474] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia, "On the uncertainty of self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 3227–3237. 180

[475] S. Lee, S. Im, S. Lin, and I. S. Kweon, "Learning Monocular Depth in Dynamic Scenes via Instance-Aware Projection Consistency," *arXiv preprint arXiv:2102.02629*, 2021. 180

[476] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei, "What's the point: Semantic segmentation with point supervision," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 549–565. 181

[477] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, and L. Van Gool, "Unsupervised Semantic Segmentation by Contrasting Object Mask Proposals," *arXiv preprint arXiv:2102.06191*, 2021. 181

[478] "Driving Computer Vision with Deep Learning," Feb. 2021. [Online]. Available: https://wayve.ai/blog/driving-computer-vision-with-deep-learning/ 181

[479] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net, 2018. 181