Learning 3D Modeling and Simulation From and For the Real World

by

Wei-Chiu Ma

B.S., National Taiwan University (2013)
M.S., Carnegie Mellon University (2016)

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2024

© 2024 Wei-Chiu Ma. All rights reserved.
The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored By: Wei-Chiu Ma
Department of Electrical Engineering and Computer Science
September 15, 2023

Certified By: Antonio Torralba
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted By: Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee for Graduate Students

To my beloved family, especially my grandparents

Learning 3D Modeling and Simulation

From and For the Real World

by

Wei-Chiu Ma

Submitted to the Department of Electrical Engineering and Computer Science
on September 15, 2023, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Humans have extraordinary capabilities of comprehending and reasoning about our 3D visual world. With just a few casual glances, we can grasp the 3D structure and appearance of our surroundings and imagine all sorts of "what-if" scenarios in our minds. Existing 3D systems, in contrast, cannot. They lack structural understanding of the world and often break apart when moved to unconstrained, partially-observed, and noisy environments. To alleviate the challenge, this thesis focus on developing robust computational tools that can effectively perceive, model, and simulate the 3D world from unconstrained sensory data. We investigate the full spectrum of dynamic 3D world understanding: from robot localization to recognition, from static 3D reconstruction to dynamic motion estimation, and from closed-loop simulation to 3D generation. By examining these tasks not only in controlled settings, but also in sparse, noisy, and sometimes even extreme real-world settings, we aim to answer the following two questions: (i) how to robustly model and reason about the visible world that we see; and (ii) how to hallucinate the unseen and imagine novel scenarios in a realistic fashion.

Thesis Supervisor: Antonio Torralba
Title: Delta Electronics Professor of Electrical Engineering and Computer Science

Acknowledgments

First, I would like to thank my PhD advisor, Antonio Torralba, for his support and guidance throughout this journey. Before joining MIT, I had heard numerous stories about Antonio's "crazy" ideas, and upon arrival, I discovered they were all true. Antonio is perhaps the most creative person I have ever met. Throughout the years, I have been constantly amazed by the ideas that he proposed or the future that he envisioned. Chatting with him is always super fun and helpful – I not only get to receive feedback on current research but also have the chance to take a peek at his grandiose vision. I will certainly miss those spontaneous interactions and chats. I also aspire for my presentations to one day be as compelling as Antonio's. Every time he steps on stage, his eyes start shining. He is always able to articulate the concept concisely and interestingly. Now, whenever I need to talk about something and am not sure what to do, I will ask myself — how would Antonio present this? And then usually I will figure the answer out. Antonio is also super kind and extremely supportive of his students. I still remember all those meetings where we simply sat down and ironed out the details of my future plans together. I cannot thank Antonio more. It is truly a great pleasure and honor to be part of the Torralba Lab. I thank Antonio for welcoming me to the family. If anyone has ever read this, please do make sure you "like" `The Great Torrabla`* fan page.

I would also like to thank Raquel Urtasun, my "unofficial" PhD co-advisor. I met Raquel eight years ago when I was a MS student, and since then she has become an incredible role model for me. She inspired me from various perspectives. Research-wise, she taught me how to rigorously approach a task and how to formulate problems in a mathematical sound fashion. I still remember when I was a junior student, Raquel would spend a lot of time guiding me through the thinking process rather than directly giving the answer away. While it was painful back then, in hindsight, the process was invaluable. It helps me to independently think and truly understand the problem. Outside of research, Raquel is a warm and caring person. She always has a lot on

*http://bit.ly/the-great-torralba

her plate (since she is a superwoman!). But no matter how busy she was, whenever I reached out to her and said I needed to chat, she was always there for me. I still remember the day when she asked me if I had her phone number and told me to call her directly if I was stressed out during my graduate studies. I was really touched. Getting to know Raquel and enjoy the ride with her is definitely one of the luckiest things that happened to me over the past few years. I cannot thank Raquel more for her tremendous support.

I am also indebted to Kris Kitani, my MS advisor at CMU. Back in 2014, I had little knowledge of computer vision. Yet Kris was willing to give me a shot and took me as one of his earlier students. I am forever grateful for that. Kris introduced me to this amazing field and taught me how to write a paper line by line. I still remember those days when Kris pulled me into his office and spent the whole afternoon teaching me how to articulate ideas clearly and concisely. We started from bullet points, sub-bullet points, and then sentences. And then, we gradually compose them back and the corresponding texts. As a junior student who had no experience in paper writing, it was extremely helpful. Kris is also an amazing human being. He always cared about his students – not just their research but also their life. I still remember all those coffee chats where we talked about all sorts of things. To me, Kris is not just a research advisor, but also a lifelong mentor. Even though I "betrayed" him and decided to pursue my Ph.D. at MIT, he still gave me his blessing and continued supporting me. Starting a research career in K-Lab is one of the most luckiest things I can ever dreamed of.

I am also grateful for my thesis committee member, Derek Hoiem and Vincent Sitzmann. I have always been a huge fan of Derek since I studied computer vision. There was a wall at CMU Smith Hall, hanging all those best papers and super popular work from the vision group. During my MS, one of my favorite things is to go through those papers and "learn from demonstrations". While I was reading them, I noticed that one name kept appearing – Derek Hoiem. Those works are so refreshing and unique and since then I have been closely following what Derek is up to. I was super excited when I got to interact with Derek more frequently since he joined my

committee last year. I really appreciate his support over the past year. I also want to thank Vincent, one of the coolest junior faculty members I have met (for reasons I will not delve into here). He is always energetic and positive. I still remember our first meeting last year. Originally, it was supposed to be a five-minute chat. Yet the conversation just kept going and at the end it was almost an hour. The discussions were super fun and I learned a lot from Vincent's unique perspectives. I thank Vincent for his support and I hope we will have the chance to collaborate in the future.

I also wanted to shout out to Sanja Fidler, whom I had the fortune to work with during my first visit to Toronto. Meeting and chatting with Sanja is always a blessing. Besides her deep knowledge in vision and graphics, Sanja is always full of energy and super optimistic. The big smile on her face always motivates me and makes me feel like everything is possible. I hope one day I can be like Sanja and provide a positive atmosphere for my research group.

I would also like to thank Shenlong Wang, who shares multiple important roles in my life: an amazing mentor, a wonderful collaborator, and a true friend. I met Shenlong eight years ago when I visited Toronto. I was a visiting student in Raquel's group and Shenlong was my mentor. Little did I know that our relationship would continue for so many years. Shenlong is an amazing role model, both inside and outside of research. He is genuine and kind to people; he is super solid and rigorous and always craving to learn more. I still remember the summer when I crashed on his couch. We discussed research on our way to the office and back home. We went swimming and secretly ate pizza without telling his wife. Although Shenlong is now busy with his professorship and his lovely family and we did not get the chance to interact as often as I would have wished to, I will always remember the joy we had.

Likewise, I am grateful to Yuxiong Wang. Yuxiong was my officemate at CMU. He is truly kind-hearted. I still remember those days when we went to grab coffee together and chat about (my) life. While Yuxiong is relatively introverted and speaks less, when he does, it is always valuable. Yuxiong has been there through various phases of my life, from my PhD application to my faculty search. His help has been instrumental – I genuinely believe I would not have gotten into MIT or landed a

faculty position without him.

I would like to pay tribute to my collaborators. During the course of my graduate studies, I have had the fortune to work with many talented minds. I have been constantly amazed by their creativity and knowledge and I have always learned a lot by chatting and discussing with them. They made me who I am today. Without their contribution, none of this thesis would have been possible. Below is a comprehensive list sorted in alphabetical order. Thank you Alberto Rodriguez Garcia, Andrei Pokrovsky, Andy Zeng, Anqi Joyce Yang, Antonio Torralba, Bin Yang, Bolei Zhou, Chen-Hsuan Lin, Chieko Asakawa, Chuang Gan, De-An Huang, Ersin Yumer, Gellert Mattyus, Hang Chu, Hang Zhao, Hao Su, Hao-Yu Max Hsu, Hironobu Takagi, Hsiao-Yu Tung, Ignacio Tartavull, Ioan Andrei Bârsan, Jack Fan, Jerry Junkai Liu, Jiayuan Gu, Jingkang Wang, Jon Barron, Joyce Anqi Yang, Justin Liang, Kris Kitani, Kaustav Kundu, Kelvin Wong, Kris Kitani, Yen-Chen Lin, Marcus Brubaker, Meet Shah, Ming Liang, Minghuan Ma, Min Bai, Namdar Homayounfar, Namhoon Lee, Pete Florence, Phillip Isola, Pranaab Dhawan, Raquel Urtasun, Rui Hu, Sanja Fidler, Shenlong Wang, Shrinidhi Lakshmikanth, Shuhan Tan, Shivam Duggal, Simon Lucey, Simon Suo, Sivabalan Manivasagam, Tatsuya Ishihara, Tian-Li Yu, Wei-Chiu Ma, Wenyuan Zeng, Xinyu Wu, Yu-Chiang Frank Wang, Yuwen Xiong, Zeng Huang, Ze Yang, Zhi-Hao Lin, Zihao Wang, and Zitian Tang.

Life during my PhD would not have been the same without my incredible labmates from the Torralba lab. My sincere thanks go to Adrià Recasens, Adrián Rodríguez Muñoz, Agata Lapedriza, Bolei Zhou, Carl Vondrick, Ching-Yao Chuang, Chuang Gan, David Bau, Dim Papadopoulos, Ethan Weber, Fern Keniston, George Cazenavette, Hang Zhao, Hengshuang Zhao, Joanna Materzynska, Jingwei Ma, Jonas Wulff, Jun-Yan Zhu, Krishna Murthy, Manel Baradad, Pratyusha Sharma, Sarah Schwettmann, Shuang Li, Steven Liu, Tamar Rott Shaham, Tianmin Shu, Tongzhou Wang, Xavier Puig Fernandez, Yichen Li, Yunzhu Li.

I also want to acknowledge the broader embodied intelligence group and the vision and graphics neighborhood for their support – Akarsh Kumar, Alex Andonian, Ali Jahanian, Andi Peng, Anthony Simeonov, Aviv Netanyahu, Bowen Pan, Boyuan Chen,

Caroline Chan, Evan Hernandez, Evan Shelhamer, Han Guo, Hyojin Bahng, Jiajun Wu, Joseph Suarez, Lucy Chai, Minyoung (Jacob) Huh, Phillip Isola, Seungwook Han, Shobhita Sundaram, Tao Chen, Tianfan Xue, Tongzhou Wang, Toru Lin, Xiuming Zhang, Yen-Chen Lin, Yonglong Tian, Zhoutong Zhang, and Abhishek Gupta.

During my time in Toronto, I had the chance to meet and interact with many outstanding researchers. Their insights have enriched my experience, and I am appreciative of everything they have taught me. My warm thanks go to Alexander Schwing, Andrei Pokrovsky, Annie Zhang, Bin Yang, Chris Zhang, Davi Frossard, Dominic Cheng, Fangyin Wei, Gellert Mattyus, Ignacio Tartavull, Ioan-Andrei Barsan, Ivan Vendrov, James Tu, Jerry Junkai Liu, Jian Yao, Jingkang Wang, Joyce Yang, Justin Liang, Kaustav Kundu, Kelvin Wong, Marcus Brubaker, Mengye Ren, Ming Liang, Min Bai, Namdar Homayounfar, Renjie Liao, Rui Hu, Sean Segal, Sergio Casas Romero, Shenlong Wang, Shrinidhi K. Lakshmikanth, Shuhan Tan, Shrivam Duggal, Simon Suo, Siva Manivasagam, Wenjie Luo, Wenjie Zi, Wenyuan Zeng, Xinchen Yan, Yuwen Xiong, Ze Yang, and Ziyu Zhang.

Life would have been much harder without the unwavering support of my friends. Their friendship has been a cornerstone outside of my academic pursuits. First, I'd like to thank the Chai Family – Lucy "Mommy" Chai and Jacob "Brother" Chai. Although they keep teasing me saying I'm over 30 and I'm old, they will become my age very soon. So we will see. I will totally miss the friendship from Ching-Yao Chuang, Yen-Chen Lin. I will never forgot those days where we keep drinking water till 3am and being super sober and being very quite in the room without disturbing the neighbors. I also wanted to thank my 2320 Eldrige roommates: Chen-Hsuan Lin, Roger Lo, and Yiuchang Lin. The two years that we have spent together laid a solid foundation for our friendship. I still clearly remember the day when they drove 12+ hours from Pittsburgh to Toronto to visit me, just because I said I felt stressed and super lonely. Even though we have been apart in different cities for years, we still constantly chat and always care for one another. They have been through all the ups and downs with me over the years. They shared my excitement when I got an interview from Antonio and participated in many of my life decisions. I do not have

brothers, but if I were to guess what it would feel like to have one, I guess it must be like this. To me, they are like my second family. For my friends in Boston, I really appreciate their company and mental support. Specifically, I thank Alexander Haojan Liu, Bin-Feng Lin, Charlotte Lin, Chen-Yu Hsu, Chieh-Chi Kao, Cynthia Chang, Edward Hsiao, Elly Cheng, Fang-Yu Liu, Hou-Yi Lin, Jessica Yang, Jin Gen Wu, Kenny Kan, Kevin Li, Po-An Tsai, Steven Huang, Tu, Wei Fang, Wei-Ning Hsu, Will Hung, Xanthe Hsu, Yiuchang Lin, and Vincent Tsao, They make Boston feel like a home to me. I will totally miss the Friday night gathering, dinner, poker, and trash talk.

Lastly, my deepest thanks go to my family. Their unwavering support and unconditional love have been my foundation. My parents have always encouraged my global pursuits. Despite my lapses in communication, they never waver in their affection. As children, my sister and I had our differences, but she is now among my closest allies. I wish my grandparents, especially my grandfathers, were here to share this moment. My niece and nephew, the newest additions to our family, may not understand this now, but I hope they read this someday and grasp the depth of my love for them.

This doctoral thesis has been examined by a Committee of the Department of Electrical Engineering and Computer Science as follows:

Professor Antonio Torralba .....
Thesis Superviser
Delta Electronics Professor of Electrical Engineering and Computer Science, MIT

Professor Raquel Urtasun .....
Member, Thesis Committee
Professor of Computer Science, University of Toronto

Professor Derek Hoiem .....
Member, Thesis Committee
Professor of Computer Science, University of Illinois at Urbana-Champaign

Professor Vincent Sitzmann .....
Member, Thesis Committee
Assistant Professor of Electrical Engineering and Computer Science, MIT

Contents

1 Introduction **27**

1.1 Dissertation Outline 29

2 Deep Structured Motion Estimation **33**

2.1 Introduction 34

2.2 Related Work 36

2.3 Deep Rigid Instance Scene Flow 38

2.3.1 Visual Cues 39

2.3.2 Energy Formulation 40

2.3.3 Inference 43

2.3.4 Learning 45

2.4 Experiments 46

2.4.1 Dataset and Implementation Details 46

2.4.2 Scene Flow Estimation 46

2.4.3 3D Rigid Motion Estimation 48

2.4.4 Analysis 50

2.5 Conclusion 52

3 Exploiting Semantics Cues and Physical Priors for Robot Localization **53**

3.1 Introduction 54

3.2 Related Work 56

3.3 Lightweight HD Mapping 59

3.4 Localization as Bayes Inference with Deep Semantics 60
3.4.1 Probabilistic Pose Filter Formulation 60
3.4.2 Efficient Inference 64
3.4.3 Learning 65
3.5 Experiments 65
3.5.1 Dataset 65
3.5.2 Implementation Details 66
3.5.3 Localization 67

4 Deep Feedback Inverse Problem Solver **71**
4.1 Introduction 72
4.2 Background 74
4.2.1 Structured optimization 75
4.2.2 Learning based methods 76
4.3 Deep Feedback Inverse Problem Solver 77
4.3.1 Deep Feedback Network 77
4.3.2 Learning 79
4.3.3 Discussions 80
4.4 Application I: 6-DoF Object Pose Estimation 83
4.5 Application II: Illumination Estimation 85
4.6 Application III: Inverse Kinematics 88
4.7 Conclusions 90

5 Exploiting High-level Priors for 3D Reconstruction **91**
5.1 Introduction 91
5.2 Related Work 94
5.3 Approach 98
5.3.1 Virtual Correspondences (VCs) 99
5.3.2 Exploiting Humans for VC Estimation 100
5.3.3 Generalized Bundle Adjustment (BA) 101
5.4 Experiments 103

5.4.1 Datasets . . . 103
5.4.2 Experimental Details . . . 104
5.4.3 Experimental Results . . . 106
5.4.4 Analysis . . . 108
5.4.5 Applications . . . 111
5.5 Conclusion . . . 112

6 Extreme 3D Modeling with Neural Radiance Fields 113
6.1 Introduction . . . 114
6.2 Related Work . . . 116
6.3 Structure from Duplicates . . . 119
6.3.1 Collaborative 6-DoF pose estimation . . . 120
6.3.2 Joint shape, material, and illumination estimation . . . 122
6.3.3 Optimization . . . 124
6.4 Experiment . . . 127
6.4.1 Experiment setups . . . 127
6.4.2 Experimental results . . . 128
6.4.3 Analysis . . . 130
6.5 Conclusion . . . 131

7 3D Simulation and Generation 133
7.1 Introduction . . . 134
7.2 Related Work . . . 137
7.2.1 Simulation . . . 137
7.2.2 Generation . . . 139
7.3 Method I: Building Editable Digital Twins . . . 140
7.3.1 Overview . . . 140
7.3.2 Preliminaries . . . 141
7.3.3 Compositional Neural Scene Representation . . . 142
7.3.4 Multi-modal Sensor Simulation . . . 144
7.3.5 Learning . . . 145

7.4 Method II: Learning Controllable Generative Models 149
7.4.1 Discrete Representations for LiDAR 149
7.4.2 LiDAR Generation 152
7.4.3 Implementation Details 153
7.5 Experiments 155
7.5.1 3D Simulation 155
7.5.2 LiDAR Scene Generation 160

8 Epilogue **165**
8.1 Future Directions 166

A Supplementary: Sparse Semantic Localization **169**
A.1 Quantitative Analysis 169
A.2 Additional Results on Hyper-Parameter Search 170

B Supplementary: Deep Structured Motion Estimation **173**
B.1 Derivation of the Solver 173
B.1.1 Gauss-Newton Solver 173
B.1.2 Jacobian of Each Energy Term 175
B.2 Impact of Unrolling Steps in GN Solver 176
B.3 Impact of energy functions 177
B.4 Curating KITTI Scene Flow 179
B.5 Qualitative Results 180
B.6 KITTI Scene Flow Benchmark 181

C Supplementary: Deep Optimizer **189**
C.1 Analyses 189
C.2 Related work 192
C.3 Other applications: JPEG image deblocking 193
C.4 Qualitative results 194

D Supplementary: Virtual Correspondences **197**

D.1 Virtual Correspondences and Epipolar Geometry 197

D.2 Bundle Adjustment for VCs 200

D.3 Quantitative Analyses 203

D.4 Implementation Details 204

D.5 Dataset Statistics 207

D.6 Qualitative Results 209

D.7 Social Impact 209

D.8 Interactive Results 213

E Supplementary: Structure from Duplicates **215**

E.1 Dataset 215

E.1.1 Synthetic data 215

E.1.2 Real-world data 215

E.2 Analyses 216

List of Figures

2-1 Performance and runtime of existing 3D motion estimators. 35

2-2 Overview of the deep structured motion estimator. 36

2-3 Qualitative evaluation of the estimated motion. 37

2-4 Qualitative comparison between different motion estimators. 41

2-5 3D rigid motion analysis. 43

2-6 Visual odometry through background motion estimation. 43

2-7 Improvement on texture-less and occluded regions. 48

3-1 Overview of the light-weight localization system. 55

3-2 Process of building semantic map. 56

3-3 Example of the inference results. 61

3-4 Qualitative results of the localization system. 70

4-1 Prior art on inverse problems. 73

4-2 Overview of deep feedback inverse problem solver. 74

4-3 Quantitative analysis on 6 DoF pose estimation. 77

4-4 Qualitative comparison on 6 DoF pose estimation. 79

4-5 Runtime breakdown of a single deep optimization step for 6 DoF pose estimation. 82

4-6 Runtime of deep optimizer vs number of faces. 82

4-7 Qualitative comparison on illumination estimation. 86

4-8 Qualitative comparison on inverse kinematics. 88

5-1 Virtual correspondences in the wild. 92

5-2 Illustration of classic correspondences and virtual correspondences. . . 95
5-3 Pipeline of estimating virtual correspondences. 95
5-4 Qualitative results of virtual correspondences. 97
5-5 Effects of camera distance on virtual correspondence estimation. . . . 98
5-6 Qualitative comparison between classic correspondences and virtual correspondences. 101
5-7 Robustness of 3D reconstruction systems with respect to camera viewpoint differences. 103
5-8 Effectiveness of 3D reconstruction systems with respect to number of input images. 107
5-9 Reconstructing scenes from extreme-view scenarios using multi-view stereo. 108
5-10 Novel view synthesis in extreme-view scenarios. 109
5-11 Estimating virtual correspondences from cars. 111

6-1 Applications of Structure *from* Duplicates. 114
6-2 Repetitions in the visual world. 116
6-3 Pipeline of Structure *from* Duplicates. 117
6-4 Qualitative comparison on multi-view inverse rendering. 125
6-5 Qualitative comparison between multi-view single object and single-view multiple instances. 126
6-6 Qualitative comparison on real-world inverse rendering. 132

7-1 Capabilities of our digital twin. 136
7-2 Capabilities of our 3D generative model. 137
7-3 Overview of our digital twin. 141
7-4 Qualitative comparison of image simulation. 147
7-5 Qualitative comparison of unconditional LiDAR generation. 151
7-6 Qualitative comparison of LiDAR simulation. 156
7-7 Qualitative comparison of Real2Sim on replay and lane shift settings. 157
7-8 Closed-loop evaluation. 159

7-9 Unconditional LiDAR generation on Pandaset. 162
7-10 Conditional LiDAR generation for dirt removal. 162

A-1 Localization error as a function of travelling distance. 169
A-2 Best hyper-parameters for each method 170
A-3 Grid search result per each hyper-parameter. 171

B-1 Performance vs unrolling steps. 177
B-2 Runtime vs unrolling steps. 178
B-3 Qualitative scene flow error map at different GN iterations. 182
B-4 Mis-alignment between the GT scene flow instance and the GT segmentation. 183
B-5 Evidence shows that some instances do not follow a rigid transform. . 183
B-6 Qualitative comparison of scene flow on test set. 184
B-7 Qualitative comparison of depth and optical flow on test set. 185
B-8 Failure cases of our motion estimation model. 186
B-9 Failure case of the inferred 3D rigid motion model. 186
B-10 Screenshot of the KITTI scene flow leaderboard at the time of paper submission. 187

C-1 Cumulative error for translation and rotation 190
C-2 Runtime w.r.t. number of faces and image size. 191
C-3 Qualitative results of 6 DoF pose estimation. 195
C-4 Qualitative results of illumination estimation. 196

D-1 Illustration of different types of virtual correspondences. 198
D-2 Comparison to classic bundle adjustment. 201
D-3 Cumulative pose error on CMU Panoptic dataset. 204
D-4 Performance vs number of images. 204
D-5 Virtual correspondences from cars. 207
D-6 Snapshot of CMU Panoptic Dataset. 208
D-7 Snapshot of Mannequin Challenge dataset. 208

D-8 Statistics of CMU Panoptic Dataset. 209
D-9 Statistics of Mannequin Challenge dataset. 209
D-10 Qualitative results on CMU Panoptic dataset. 210
D-11 Qualitative Results on Mannequin Challenge dataset. 211
D-12 Qualitative Results on movies, sitcom, and sports photograph. 212

E-1 Single-view multi-instances vs multi-view single instance. 216
E-2 Importance of rotation augmentation for structure from motion. . . . 217
E-3 Single-view inverse rendering of the "cleaner" scene. 218
E-4 Single-view inverse rendering of the "cash machine" scene. 219

List of Tables

2.1 Quantitative comparison against prior art on motion estimation. . . . 39

2.2 Contributions of each energy on motion estimation. 46

2.3 Improvement over original flow/stereo estimation. 47

2.4 Runtime analysis of deep structured motion estimator. 47

3.1 Quantitative comparison on localization accuracy. 68

3.2 Quantitative comparison on smoothness. 69

3.3 Contribution of each component on localization. 69

3.4 Effectiveness of probabilistic filtering. 70

4.1 Quantitative comparison on 6 DoF pose estimation. 78

4.2 Generalization of deep optimizer on 6 DoF pose estimation. 83

4.3 Quantitative comparison on illumination estimation. 85

4.4 Quantitative comparison on inverse kinematics. 88

5.1 Quantitative evaluation on two frame camera pose estimation. 105

5.2 Effectiveness of different types of correspondences. 106

6.1 Quantitative evaluation on synthetic multi-view inverse rendering. . . 119

6.2 Quantitative evaluation on synthetic single-view inverse rendering. . . 119

6.3 Performance of structure from duplicates with respect to the number of instances. 122

6.4 Quantitative evaluation on real-world single-view inverse rendering. . 122

6.5 Quantitative comparison between multi-view single object and single-view multiple instances. 129

6.6 Effectiveness of collaborative 6 DoF pose estimation. 129

7.1 Quantitative comparison of image simulation. 147
7.2 Contribution of each component on simulation 148
7.3 Quantitative comparison of LiDAR simulation. 148
7.4 Detection domain gap. 157
7.5 Data augmentation via image simulation. 158
7.6 Open-Loop Real2Sim Autonomy Evaluation 158
7.7 Quantitative comparison of LiDAR generation. 161
7.8 Human evaluation on LiDAR generation. 161

B.1 Contributions of each energy. 179

C.1 Deep feedback inverse problem solver as an initializer. 190
C.2 Effectiveness of unrolling steps. 190
C.3 Effectiveness of not sharing weight across stages. 191
C.4 Effectiveness of image size on 6 DoF pose estimation. 192

D.1 Pose estimation on CMU Panoptic dataset. 205
D.2 Pose estimation on Mannequin Challenge dataset. 205

Chapter 1

Introduction

Demand for robust and automated reasoning of our three-dimensional (3D) physical world is higher than ever. To navigate traffic safely, autonomous vehicles must perceive their surroundings and predict how it will evolve over time. Augmented reality (AR) and virtual reality (VR) systems must understand the scene's geometry, material, and lighting to manipulate the scene and produce new photorealistic visual content. Personal robots need to recognize the poses and affordances of everyday objects to interact with them effectively. Unfortunately, despite substantial recent progress on 3D modeling and simulation, most algorithms still assume relatively controlled setups (*e.g.*, assuming well-calibrated poses or dense overlapping images) or require a large amount of supervision to span all scenarios. With limited data, they fail to generalize to *unconstrained, partially-observed, noisy* environments often encountered by real-world applications.

Indeed, data captured in real-world settings presents a set of unique difficulties when compared to data designed in the lab. The sensory data (*e.g.*, images, LiDAR point clouds) *varies* significantly (*e.g.*, illumination change, point density difference) and are often *noisy* (*e.g.*, due to weather, motion); the observations are usually *sparse* and only provide incomplete views of the world. These factors pose severe challenges to existing machinery. Interestingly, even under such challenging conditions, humans can effortlessly exploit priors for 3D modeling and reasoning to make good predictions. With just a few casual glances, we can grasp the 3D structure and appearance of our

surroundings and imagine all sorts of "what-if" scenarios. Such a capability, while intrinsic to humans, remains challenging for state-of-the-art computational models.

The goal of this thesis is to equip intelligent systems with similar abilities. We aim to develop computational tools that can effectively *perceive, model,* and *simulate* the 3D world from unconstrained sensory data. Towards this goal, we first study how to *combine the flexibility of deep neural networks with structured inductive biases*, which allows us to significantly improve the performance, robustness, computational efficiency, and data dependency across a variety of 3D tasks and generalize to in-the-wild variation with minimal amounts of information. Our key insight is to leverage geometric and physical structure into learning frameworks. By integrating top-down structural reasoning (*e.g.*, optimization) with bottom-up computational models (*e.g.*, deep networks), our models become more robust to data variation, less data-hungry, and, most importantly, can guarantee physically-plausible results during inference. This is crucial to applications such as self-driving since it is impossible to cover all scenarios during training, and the estimated output (*e.g.*, free space, localization, 3D motion) will be fed into downstream physical-based planners and controllers.

Next, we explore how to construct *composable, editable,* and *actionable* 3D representations that allow robotic systems (*e.g.*, self-driving vehicles, robot arms) to simulate counterfactual scenarios for better decision-making. By realistically simulating temporally and spatially consistent sensory data (*e.g.*, images, LiDAR point clouds) at novel viewpoints and for different scene configurations (*e.g.*, actors at new placements), we can generate potentially infinite synthetic yet realistic training data for machine learning models and test the policy of autonomous systems on a variety of scenarios, including hazardous long-tail situations that are difficult to test safely, without needing to deploy to the real world.

We investigates the full spectrum of dynamic 3D world understanding: from robot localization to recognition, from static 3D reconstruction to dynamic motion estimation, and from closed-loop simulation to 3D generation. We examine these tasks not only in controlled settings, but also in sparse, noisy, and sometimes even extreme real-world settings where the models will be deployed in. In the following, we will

present our research contributions along two main axes: (i) robust and efficient 3D reasoning in the wild, and (ii) building digital replicas of the world.

1.1 Dissertation Outline

The remainder of the thesis is organized as follows. We start by discussing how to robustly model and reason about the visible world, addressing a wide array of 3D challenges. Then, we delve deeper, presenting methods to hallucinate the unseen and imagine new scenarios with a touch of realism. More specifically:

– Chapter 2 investigates the rigidity property in 3D motion estimation. We find that by integrating deep learning techniques with strong priors specific to our application domain – wherein the scene's motion is a composite of both the robot's movement and the 3D motion of the actors within the scene – we can effectively and robustly recover the underlying 3D motion. Specifically, we frame the problem as an energy minimization task within a deep structured model. The formulation allows us to solve the problem efficiently on the modern GPU. Compared to the state-of-the-art approaches, our method achieves superior performance while being 800 times faster.

– Chapter 3 investigates the importance of semantics within robot localization tasks and proposes a novel, lightweight semantic localization algorithm that exploits multiple sensors and has precision on the order of a few centimeters. We find that exploiting semantic cues such as lanes and traffic signs, together with vehicle dynamics, is sufficient to localize a self-driving vehicle robustly with respect to a sparse semantic map. This significantly reduces the need for detailed knowledge about the world's appearance in advance. Additionally, it requires significantly less storage than maps utilized by traditional geometry- and LiDAR intensity-based localizers. We validate the effectiveness of our method on a new highway dataset and demonstrate superior performance while taking up only a fraction of memory.

– Chapter 4 investigates how to leverage physical processes as supervisory training signals and equip models with better understanding of our world. We focus on inverse problems where the forward process is well-defined and leverage the feedback signal provided by the physical forward process to learn an iterative update model. Through the feedback information, our model not only can produce accurate estimations that are coherent to the input observation but also is capable of recovering from early incorrect predictions. On various inverse problems, our method achieves comparable or better performance while being two to three orders of magnitude faster than traditional approaches.

– Chapter 5 explores the potential of using prior knowledge of the objects within the scene for extreme-view 3D reconstruction. In particular, we study how to combine shape priors from recognition-based single-view methods with geometry-based multi-view methods. We find that by hallucinating what objects would look like from other viewpoints and then matching them with the visible regions from other images, we can associate the pixels geometrically even if they have completely different semantics and appearances. We further incorporate these new correspondences into classic 3D reconstruction pipelines and demonstrate significantly better performance in challenging scenarios and comparable results in the traditional setup.

– Chapter 6 further explores our knowledge of the world to reason about both the underlying geometric and physical properties under extreme conditions. Based on the observations that our world is full of identical objects (*e.g.*, cans of coke, cars of the same model) and these duplicates, when seen together, provide additional and strong cues for us to effectively reason about 3D properties, we introduce a novel inverse graphics framework that reconstructs geometry, material, and illumination from a single image containing multiple identical objects. By utilizing repetitive patterns in images to extract and utilize multi-view information, we can significantly outperform existing single-image reconstruction models and multi-view reconstruction approaches with a similar or greater number of

observations.

– Chapter 7 describes our efforts to build editable, composable, and interpretable digital twins of the 3D world. We draw inspiration from the latest advances in neural rendering and generative modeling and present two distinct approaches that can not only faithfully reproduce what it "sees," but also synthesize novel, high-fidelity observations of the world. We test our systems on a wide variety of downstream tasks and also conduct human evaluation. Experiments show that our synthesized results are much more realistic, exhibit a significantly reduced domain gap, and unlock the potential of training and evaluation of autonomous systems in simulation.

Chapter 2

Deep Structured Motion Estimation

Deep Rigid Instance Scene Flow

Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, Raquel Urtasun;
CVPR 2019.

We start by exploring how infusing structural priors into computational models can improve the overall robustness, efficiency, and performance of various intelligent 3D systems. We begin with dynamic motion estimation, a fundamental and critical building block for understanding the dyanmics and evolvement of the 3D scene. Since in-the-wild motion data is often scarce and noisy, naively applying deep learning will result in infeasible estimations where different parts of an object will move inconsistently. We thus explicitly encode various constraints (*e.g.* 2D-3D consistency) into a neural network to ensure that all parts move as a whole. The result is a differentiable and end-to-end learnable pipeline, enabling improved performance while guaranteeing physically-feasible output. It was one of the earliest works that incorporated optimization processes into deep learning architectures for 3D reasoning. This framework has set the stage for follow-up work on scene flow estimation, and has inspired other optimization-inspired 3D neural networks.

2.1 Introduction

Scene flow refers to the problem of estimating a three-dimenional motion field from a set of two consecutive (in time) stereo pairs. It was first introduced in [355] to describe the 3D motion of each point in the scene. Through scene flow, we can gain insights into the geometry as well as the overall composition and motion of the scene. It is of particular importance for robotics systems, such as self-driving cars, as knowing the 3D motion of other objects in the scene can not only help the autonomous systems avoid collision while planing its own future movements, but also improve the understanding of the scene and predict the intent of others. In this work, we focus on estimating the 3D scene flow in autonomous driving scenarios.

In the world of self-driving, the motion of the scene can be mostly explained by the motion of the ego-car. The presence of dynamic objects which typically move rigidly can also be utilized as strong priors. Previous structure prediction approaches often exploit these facts and fit a piece-wise rigid representations of motion [358, 397, 243, 27]. While these methods achieve impressive results on scene flow estimation, they require minutes to process each frame, and thus cannot be employed in real-world robotics systems.

On the other hand, deep learning based methods have achieved state-of-the-art performance in real time on a variety of low level tasks, such as optical flow prediction [79, 288, 332] and stereo estimation [416, 242, 227]. While they produce 'accurate' results, their output is not structured and cannot capture the relationships between estimated variables. For instance, they lack the ability to guarantee that pixels on a given object produce consistent estimates. While this phenomenon may have little impact in photography editing applications, this can cathastrophic in the context of self-driving cars, where the motion of the full object is more important than the motion of each individual pixel.

With these problems in mind, we develop a novel deep rigid instance scene flow (DR-ISF) model that takes the best of both worlds. The idea behind is that the motion of the scene can be composed by estimating the 3D rigid motion of each actor. The

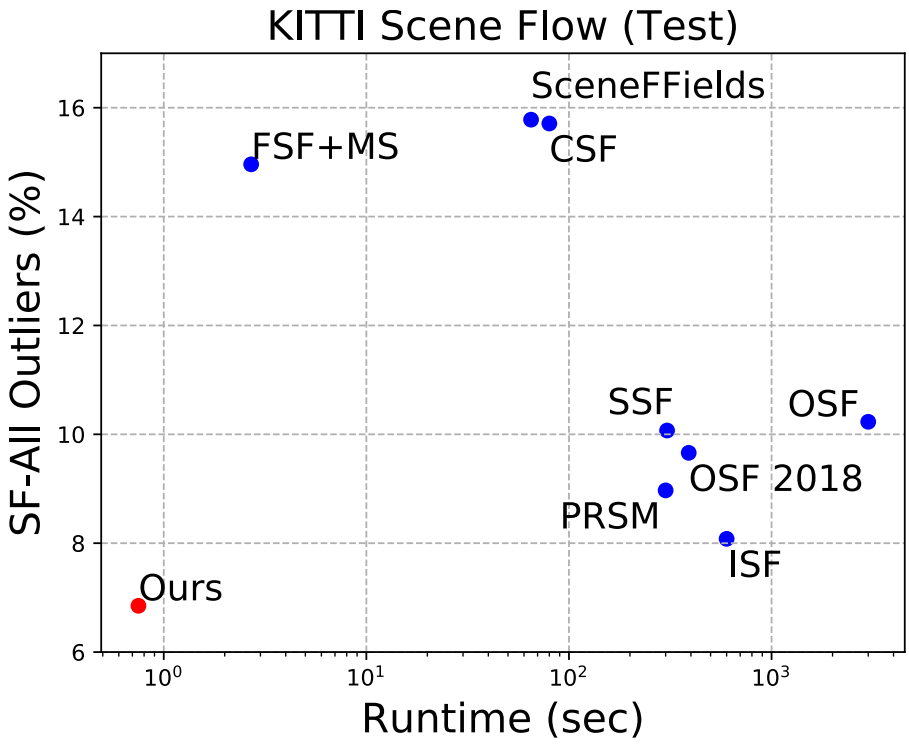


Figure 2-1: **Performance vs runtime on KITTI SceneFlow dataset:** Our approach is much faster and more accurate.

static background can also be modeled as a rigidly moving object, as its 3D motion can be described by the 'ego-car' motion. The problem is thus reduced to estimating the 3D motion of each traffic participant. Towards this gaol, we first capitalize on deep neural networks to estimate optical flow, disparity and instance segmentation. We then exploit multiple geometry based energy functions to encode the structural geometric relationship between these visual cues. Through optimizing the energy function, we can effectively reason about the 3D motion of each traffic participant. As the energy takes the form of weighted sum of squares, it can be efficiently minimized via Gaussian-Newton (GN) algorithm [33]. We implement the GN solver as layers in neural networks, thus all operations can be computed efficiently on the GPU in an end-to-end fashion.

We demonstrate the effectiveness of our approach on the KITTI scene flow dataset [243]. As shown in Fig. 2-1, our deep rigid instance scene flow model outperforms all previous methods by a significant margin in both runtime and accuracy. Importantly, it achieves state-of-the-art performance on almost every entry. Comparing to prior

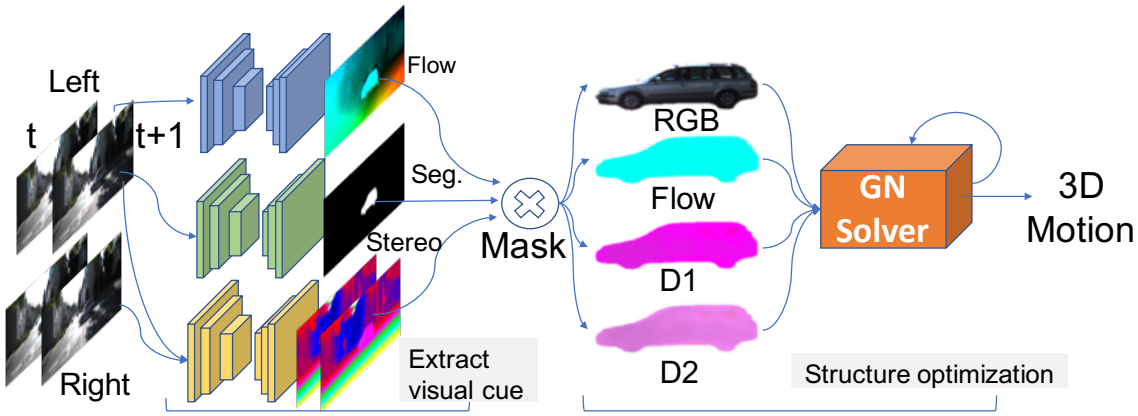


Figure 2-2: **Overview of our approach:** Given two consecutive stereo images, we first estimate the flow, stereo, and segmentation (Sec. 2.3.1). The visual cues of each instance are then encoded as energy functions (Sec. 2.3.2) and passed into the Gaussian-Newton (GN) solver to find the best 3D rigid motion (Sec. 2.3.3). The GN solver is unrolled as a recurrent network.

art, DRISF reduces the D1 outliers ratio by **43%**, the D2 outliers ratio by **32%**, and the flow outliers ratio by **24%**. Comparing to the existing best scene flow model [27], our scene flow error is **22%** lower and our runtime is **800** times faster.

2.2 Related Work

Optical flow: Optical flow is traditionally posed as an energy minimization task. It dates back to Horn and Schunck [138] where they define the energy as a combination of a data term and a smoothness term, and adopt variational inference to solve it. Since then, a variety of improvements have been proposed [36, 31, 269]. Recently, deep learning has replaced the variational approaches. Employing deep features for matching [15, 369] improved performance by a large margin. However, as the matching results are not dense, post-processing steps are required [296]. This not only reduces the speed, but also limits the overall performance.

Pioneered by Flownet [79], various end-to-end deep regression based methods have been proposed [147]. Flownet2 [146] stacks multiple networks to iteratively refine the estimated flow and introduces a differentiable warping operation to compensate for large displacements. As the resulting network is very large, SpyNet [288] propose to

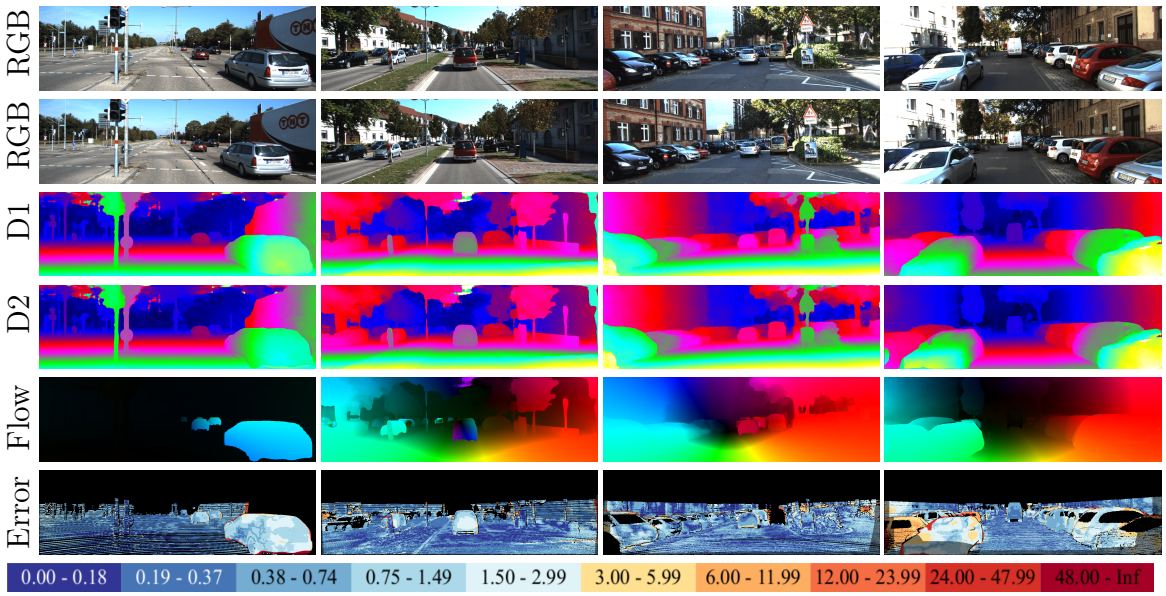


Figure 2-3: **Qualitative results on validation set:** Our model can estimate the background motion very accurately. It is also able to estimate the 3D motion of foreground objects in most scenarios. It fails in challenging cases as show in last column.

use spatial pyramid network to handle large motions. They reduce the model size greatly, yet at the cost of degrading performance. Lite-Flownet [143] and PWC-Net [332, 333] extend this idea and incorporate the traditional pyramid processing and cost volume concepts into the network. Comparing to previous approach, the resulting model is smaller and faster. In this work, we adapt the latest PWC-Net as our flow module.

Stereo: Traditional stereo methods [135, 161] follow three steps: compute patch-wise feature, construct cost volumes, and final post-processing. The representation of the patch plays an important role. Modern approaches leverage CNNs to predict whether two patches are a match [413, 416]. While they showed great performance in challenging benchmarks, they are computationally expensive. To speed up the matching process, Luo *et al.* [227] propose a siamese matching network which exploits a correlation layer [60] to extract marginal distributions over all possible disparities. While the usage of the correlation layer significantly improves efficiency, they still require post-processing techniques [133, 417] to smooth their estimation, which largely

limits their speed. In light of this, networks that directly regress sub-pixel disparities from the given stereo image pair have been proposed. DispNet [242] exploits a 1D correlation layer to approximate the stereo cost volumes and rely on later layers for implicit aggregation. Kendall *et al*. [170] incorporate 3D conv for further regularization and propose a differentiable soft argmin to enable sub-pixel disparity from cost volumes. PSM-Net [53] later extend [170] by incorporating stacked hourglass [257] and Pyramid spatial pooling [433, 127]. In this work, we exploit PSM-Net as our stereo module.

Scene flow: Scene flow [355] characterizes the 3D motion of a point. Similar to optical flow estimation, the task is traditionally formulated as a variational inference problem [353, 276, 142, 24]. However, the performance is rather limited in real world scenarios due to errors caused by large motions. To improve the robustness, slanted-plane based methods [397, 243, 358, 228] propose to decompose the scene into small rigidly moving planes and solve the discrete-continuous optimization problem. Behl *et al*. [27] build upon [243], and incorporate recognition cues. With the help of fine-grained instance and geometric feature, they are able to establish correspondences across various challenging scenarios. Similar to our work, Ren *et al*. [295] exploit multiple visual cues for scene flow estimation. They encode the features via a cascade of conditional random fields and iteratively refine them. While these methods have achieved impressive performance, they are computationally expensive for practical usage. Most methods require minutes to compute one scene flow. This is largely due to the complicated optimization task. In contrast, our deep structured motion estimation model is able to compute scene flow in less than a second, which is two to three orders of magnitude faster.

2.3 Deep Rigid Instance Scene Flow

In this chapter we are interested in estimating scene flow in the context of self-driving cars. We build our model on the intuition that in this scenario the motion of the scene can be formed by estimating the 3D motion of each actor. The static background can

		Dispairty 1			Dispairty 2			Optical Flow			Scene Flow		
Methods	Runtime	*bg*	*fg*	all	*bg*	*fg*	all	*bg*	*fg*	all	*bg*	*fg*	all
CSF [228]	1.3 mins	4.57	13.04	5.98	7.92	20.76	10.06	10.40	25.78	12.96	12.21	33.21	15.71
OSF [243]	50 mins	4.54	12.03	5.79	5.45	19.41	7.77	5.62	18.92	7.83	7.01	26.34	10.23
SSF [295]	5 mins	3.55	8.75	4.42	4.94	17.48	7.02	5.63	14.71	7.14	7.18	24.58	10.07
OSF-TC* [256]	50 mins	4.11	9.64	5.03	5.18	15.12	6.84	5.76	13.31	7.02	7.08	20.03	9.23
PRSM* [359]	5 mins	3.02	10.52	4.27	5.13	15.11	6.79	5.33	13.40	6.68	6.61	20.79	8.97
ISF [27]	10 mins	4.12	6.17	4.46	4.88	11.34	5.95	5.40	**10.29**	6.22	6.58	**15.63**	8.08
Our DRISF	**0.75 sec**	**2.16**	**4.49**	**2.55**	**2.90**	**9.73**	**4.04**	**3.59**	10.40	**4.73**	**4.39**	15.94	**6.31**

Table 2.1: **Comparison against top 6 published approaches:** Our method achieves state-of-the-art performance on almost every entry while being two to three orders of magnitude faster. (*: Method uses more than two temporally adjacent images.)

be also modeled as a rigidly moving object, as its 3D motion can be described by the 'ego-car' motion. Towards this goal, we proposed a novel deep structured model that exploits optical flow, stereo, as well as instance segmentation as visual cues. We start by describing how we employ deep learning to effectively estimate the geometric and semantic features. We then formulate the scene flow task as an energy minimization problem and discuss each energy term in details. Finally, we describe how to perform efficient inference and learning.

2.3.1 Visual Cues

We exploit three types of visual cues: instance segmentation, optical flow and stereo.

Instance Segmentation: We utilize Mask R-CNN [128] as our instance segmentation network, as it produces state-of-the-art results in autonomous driving benchmarks such as KITTI [102] and Cityscapes [66]. Mask R-CNN is a proposal based two stage network built upon Faster R-CNN [294]. For each object proposal, it predicts the object class, regresses its 2D box, and infers the bg/fg segmentation mask.

Stereo: We exploit the pyramid stereo matching network (PSM-Net) [53] to compute our stereo estimates. It consists of three main modules: fully convolutional feature module, spatial pyramid pooling [127, 433] and 3D cost volume processing. The feature module computes a high-dimensional feature map in a fully convolutional manner; the spatial pyramid pooling aggregates context in different scales and locations to

construct the cost volume; the 3D cost volume module then performs implicit cost volume aggregation and regularizes it using stacked hourglass networks. Compared to previous disparity regression networks, PSM-Net learns to refine and produce sharp disparity images that respect object boundaries better. This is of crucial importance as over-smoothed results often deteriorates motion estimation.

Optical Flow: Our flow module is akin to PWC-Net [332], which is a state-of-the-art flow network designed based on three classical principles (similar to stereo networks): pyramidal feature processing, warping, and cost volume reasoning. Pyramidal feature processing encode visual features with large context; the progressive warping reduces the cost of building cost-volume through a coarse-to-fine scheme. Cost volume reasoning further boost performance by sharpening the boundaries. We implement PWC-net with one modification: during the warping operation, we use the feature of the nearest boundary pixel to pad if the sampling point falls outside the image, rather than 0. Empirically we found this to improve performance.

2.3.2 Energy Formulation

We now describe the energy formulation of our deep structured model. Let $\mathcal{L}^0, \mathcal{R}^0, \mathcal{L}^1, \mathcal{R}^1$ be the input stereo pairs captured from two consecutive time steps. Let $\mathcal{D}^0, \mathcal{D}^1$ be the estimated stereo, and $\mathcal{F}_\mathcal{L}, \mathcal{F}_\mathcal{R}$ be the inferred flow. Denote $\mathcal{S}^0_\mathcal{L}$ as the instance segmentation computed on the left image $\mathcal{L}^0$. Assume all cameras are pre-calibrated with known intrinsics. We parametrize the 3D rigid motion with $\boldsymbol{\xi} \in \mathfrak{se}(3)$, the Lie-algebra associated with $SE(3)$. We use this parametrization as it is a minimal representation for 3D motion. For each instance $i \in \mathcal{S}^0_\mathcal{L}$, we aim to find the rigid 3D motion that minimizes the weighted combination of photometric error, rigid fitting and flow consistency, where the weights are denoted as $\lambda_{\cdot,i}$. For simplicity, let $\mathcal{I} = \{\mathcal{L}^0, \mathcal{R}^0, \mathcal{L}^1, \mathcal{R}^1, \mathcal{D}^0, \mathcal{D}^1, \mathcal{F}_\mathcal{L}, \mathcal{F}_\mathcal{R}\}$ be input images and visual cues. We denote the set of pixels belonging to instance i as $P_i = \{\boldsymbol{p} | S^0_\mathcal{L}(\boldsymbol{p}) = i\}$. Note that background can be considered as an 'instance' since all the pixels in it undergo the same rigid

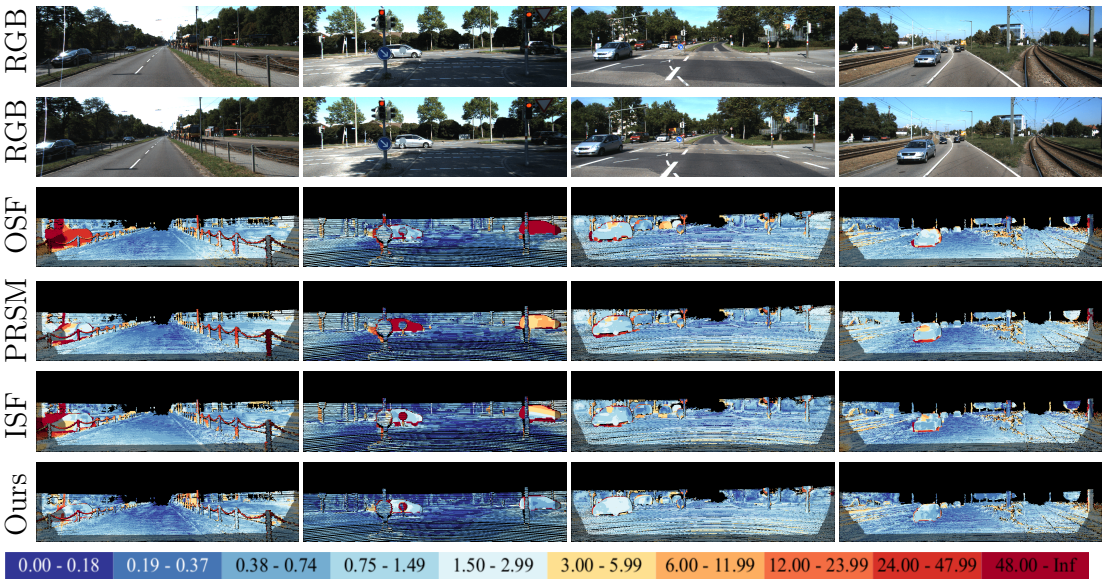


Figure 2-4: **Qualitative comparison on test set:** Our method can effectively handle occlusion and texture-less regions. It is more robust to the illumination change as well as large displacement. Please refer to the supp. material for more results.

transform. We obtain the 3D motion of each instance by minimizing

$$\min_{\boldsymbol{\xi}} \{\lambda_{\text{photo},i} E_{\text{photo},i}(\boldsymbol{\xi}; \mathcal{I}) + \lambda_{\text{rigid},i} E_{\text{rigid},i}(\boldsymbol{\xi}; \mathcal{I}) + \lambda_{\text{flow},i} E_{\text{flow},i}(\boldsymbol{\xi}; \mathcal{I})\} \tag{2.1}$$

The three energy terms are complementary. They capture the geometry and appearance agreement between the observations and inferred rigid motion. Next, we describe the energy terms in more details.

Photometric Error: This energy encodes the fact that correspondences should have similar appearance across all images. In particular, for each pixel $\boldsymbol{p} \in P_i$ in the reference image, we compare its photometric value with that of the corresponding pixel in the target image:

$$E_{\text{photo},i}(\boldsymbol{\xi}; \mathcal{I}) = \sum_{\boldsymbol{p} \in P_i} \alpha_{\boldsymbol{p}} \rho(\mathcal{L}^0(\boldsymbol{p}) - \mathcal{L}^1(\boldsymbol{p}')) \tag{2.2}$$

where $\alpha_{\boldsymbol{p}} \in \{0, 1\}$ is an indicator function representing which pixel is an outlier. We refer the reader to section 2.3.3 for a discussion on how to estimate α_p. $\boldsymbol{p}$ is a pixel in the reference image and $\boldsymbol{p}'$ stands for the projected image coordinate on another image, given by inverse depth warping followed by a rigid transform $\boldsymbol{\xi}$. Specifically,

$$\boldsymbol{p}' = \pi_{\mathbf{K}}(\boldsymbol{\xi} \circ \pi_{\mathbf{K}}^{-1}(\boldsymbol{p}, \mathcal{D}(\boldsymbol{p}))) \tag{2.3}$$

where $\pi_{\mathbf{K}}(\cdot) : \mathbb{R}^3 \to \mathbb{R}^2$ is the perspective projection function given known intrinsic $\mathbf{K}$ and $\pi_{\mathbf{K}}^{-1}(\cdot, \cdot) : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$ is the inverse projection that convert a pixel and its associated disparity into a 3D point; $\boldsymbol{\xi} \circ \mathbf{x}$ transforms a 3D point $\mathbf{x}$ rigidly with transformation $\exp(\boldsymbol{\xi})\mathbf{x}$. ρ is a robust error function that improves the overall robustness by reducing the influence of outliers on the non-linear least squares problems. Following Sun *et al.* [331], we adopt the generalized Charbonnier function $\rho(x) = (x^2 + \epsilon^2)^{\alpha}$ as our robust function and set $\alpha = 0.45$ and $\epsilon = 10^{-5}$. Similar to [331], we observe the slightly non-convex penalty improves the performance in practice.

Rigid Fitting: This term encourages the estimated 3D rigid motion to be similar to the point-wise 3D motion obtained from the stereo and flow networks. Formally, given correspondences $\{(\boldsymbol{p}, \boldsymbol{q} = \boldsymbol{p} + F_{\mathcal{L}}(\boldsymbol{p}))|\boldsymbol{p} \in P_i\}$ defined by the output of optical flow network and the disparity maps $\mathcal{D}^0, \mathcal{D}^1$, the energy measures rigid fitting error of $\boldsymbol{\xi}$:

$$E_{\text{rigid},i}(\boldsymbol{\xi}; \mathcal{I}) = \sum_{(\boldsymbol{p},\boldsymbol{q})} \alpha_{\boldsymbol{p}} \rho(\boldsymbol{\xi} \circ \pi_{\mathbf{K}}^{-1}\left(\boldsymbol{p}, \mathcal{D}^0(\boldsymbol{p})\right) - \pi_{\mathbf{K}}^{-1}\left(\boldsymbol{q}, \mathcal{D}^1(\boldsymbol{q})\right)),$$

where $\boldsymbol{q} = \boldsymbol{p} + \mathcal{F}_{\mathcal{L}}(\boldsymbol{p})$ and $\pi_{\mathbf{K}}^{-1}$ denotes the inverse projection function, and ρ is the same robust error function.

Flow Consistency: This term encourages the projection of the 3D rigid motion to be close to the original flow estimation. This is achieved by measuring the difference between our optical flow net, and the structured rigid flow, which is computed by

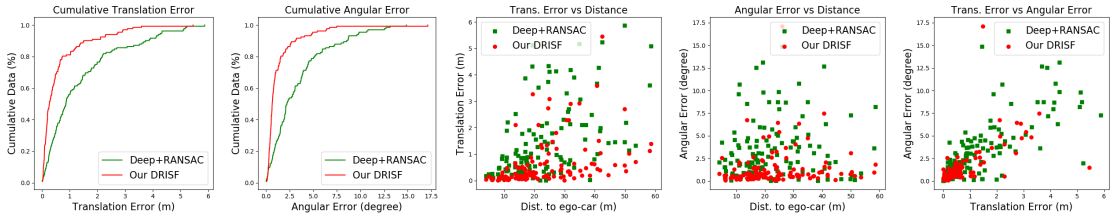


Figure 2-5: **3D rigid motion analysis:** Over 80% of the estimated 3D rigid motion has an error less than $1m$ and $1.3°$. Large errors often happen at farther distances where the vehicles are small and less points are observable.

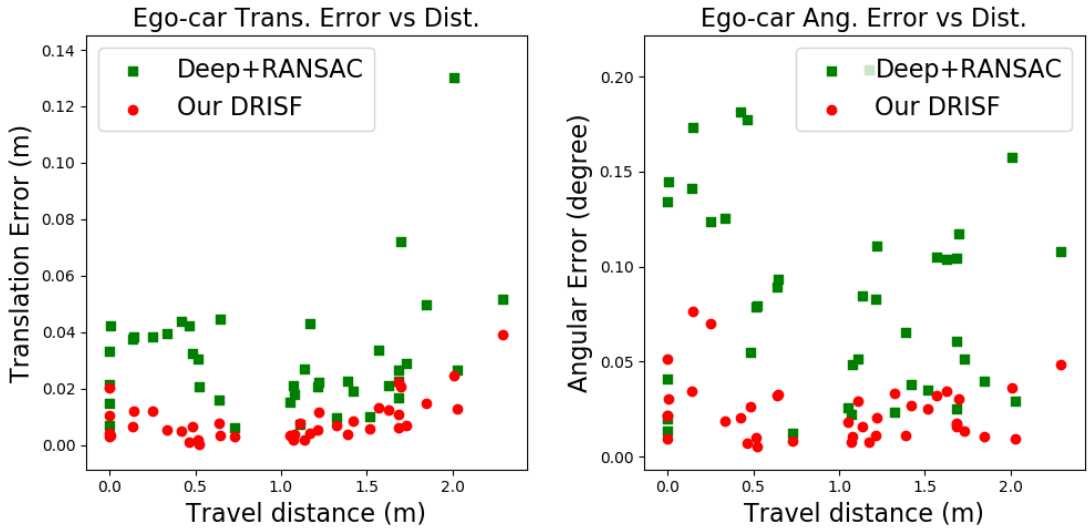


Figure 2-6: **Odometry from background motion:** On average, our ego-car drifts $0.9cm$ and $0.024°$ every $1m$ of drive.

warping each pixel using $\mathcal{D}^0$ and the rigid motion $\boldsymbol{\xi}$.

$$E_{\text{flow},i}(\boldsymbol{\xi};\mathcal{I}) = \sum_{\boldsymbol{p}\in P_i} \rho\Big(\underbrace{(\boldsymbol{p}' - \boldsymbol{p})}_{\text{2D Rigid flow}} - \underbrace{\mathcal{F}_{\mathcal{L}}(\boldsymbol{p})}_{\text{optical flow}} \Big) \tag{2.4}$$

where $\boldsymbol{p}'$ is the rigid warping function defined in Eq. (2.3), and ρ is the same robust error function.

2.3.3 Inference

Uncertain Pixel Removal: Due to viewpoint change, flow/stereo prediction errors, etc, the visual cues of some pixels are not reliable. For instance, pixels in one image

may be occluded in another image due to viewpoint change. This motivates us to assign $\alpha_{\boldsymbol{p}}$ to each pixel $\boldsymbol{p}$ as an indication of outlier or not. Towards this goal, we first exclude pixels which are likely to be occluded in the next frame. Specifically, pixels are labeled as occluded if the warped 3D disparity of the second frame significantly differs from the disparity of the first frame. The intuition is that the disparity of a pixel cannot change drastically in real world due to the speed limit. We empirically set threshold to 30. Next, we employ the RANSAC scheme to fit a rigid motion for each instance. We only keep the inlier points and prune out the rest. Despite simple, we found this strategy very effective.

Initialization: Due to the highly non-convex structure of the energy model, a good initialization is critical to achieve good performance. As previous step already prune out most unreliable points, we directly exploit the rigid motion obtained by RANSAC as our robust initial guess.

Gaussian Newton Solver: The energy function is non-convex but differentiable w.r.t. $\boldsymbol{\xi}$ defined over continuous space. In order to handle the robust function, we adopt an iterative re-weighted least square algorithm [43]. For each iteration, we can rewrite the original energy minimization problem of each instance i as a weighted sum of squares:

$$\boldsymbol{\xi}^{(n+1)} = \arg\min_{\boldsymbol{\xi}} E_{\text{total},i}(\boldsymbol{\xi}) = \arg\min_{\boldsymbol{\xi}} \sum_{\text{Eng}} w_i(\boldsymbol{\xi}^{(n)}) r_i^2(\boldsymbol{\xi}^{(n)}),$$

where r denotes the residual function, w re-weights each sample based on the robust function ρ, and Eng refers to summing over the energy terms. We employ Gaussian-Newton algorithm to minimize the function. Thus we have

$$\boldsymbol{\xi}^{(n+1)} = -(\mathbf{J}^T\mathbf{W}\mathbf{J})^{(-1)}\mathbf{J}^T\mathbf{W}r(\boldsymbol{\xi}^{(n)}) \circ \boldsymbol{\xi}^{(n)} \tag{2.5}$$

where $\circ$ is a pose composition operator and $\mathbf{J} = \frac{\delta r(\boldsymbol{\epsilon} \circ \boldsymbol{\xi}^{(n)})}{\delta \boldsymbol{\epsilon}}|_{\boldsymbol{\epsilon}=0}$. In practice, we unroll the inference steps as a recurrent neural network and define its computation graph as

in Eq. (2.5). The full pipeline including the matrix inverse is differentiable. Please refer to the supp. material for the derivation of the Jacobian matrix of each term and more details on the Gaussian-Newton solver.

Final Scene Flow Prediction: Given the final rigid motion estimation for each instance $\boldsymbol{\xi}_i^*$, we are able to compute the dense instance-wise rigid scene flow. Our scene flow consists of three component, namely the first frame's stereo $\mathcal{D}^0$, warped stereo to second frame $\mathcal{D}^{\text{warp}}$ as well as the instance-wise rigid flow estimation $\mathcal{F}^{\text{rigid}}$. Specifically, for each point $\boldsymbol{p}$ we have:

$$\begin{aligned} \mathcal{D}^0(\boldsymbol{p}) &= \mathcal{D}^0(\boldsymbol{p}) \\ \mathcal{D}^{\text{warp}}(\boldsymbol{p}) &= z_{\mathbf{K}}(\boldsymbol{\xi}^*_{\mathcal{S}^0_{\mathcal{L}}(\boldsymbol{p})} \circ \pi^{-1}_{\mathbf{K}}(\boldsymbol{p}, \mathcal{D}^0(\boldsymbol{p}))) \\ \mathcal{F}^{\text{rigid}}(\boldsymbol{p}) &= \boldsymbol{p}' - \boldsymbol{p} = \pi_{\mathbf{K}}(\boldsymbol{\xi} \circ \pi^{-1}_{\mathbf{K}}\left(\boldsymbol{p}, \mathcal{D}^0(\boldsymbol{p})\right)) - \boldsymbol{p} \end{aligned} \tag{2.6}$$

where $z_{\mathbf{K}}(\cdot)$ computes the disparity of the 3D point; $\pi^{-1}_{\mathbf{K}}$ is the inverse projection function; and $\boldsymbol{\xi} \circ \mathbf{x}$ transforms a 3D point $\mathbf{x}$ using the rigid motion $\boldsymbol{\xi}$.

2.3.4 Learning

The whole deep structured network can be trained end-to-end. In practice, we train our instance segmentation, flow estimation, and stereo estimation module respectively through back-propagation. To be more specific, Mask R-CNN model is pre-trained on Cityscapes and fine-tuned on KITTI. The loss function includes ROI classification loss, box regression loss as well as the mask segmentation loss. PSM-Net is pre-trained on Scene Flow [242] and fine-tuned on KITTI with L1 regression loss. PWC-Net is pre-trained on FlyingChairs [79] and FlyingThings [242] then fine-tuned over KITTI, with weighted L1 regression loss.

Employed energy			Background outliers (%)				Employed energy			Foreground outliers (%)			
E_{pho}	E_{flow}	E_{rigid}	D1	D2	Fl	SF	E_{pho}	E_{flow}	E_{rigid}	D1	D2	Fl	SF
✓			1.92	2.69	**3.71**	**4.30**	✓			1.70	**4.25**	7.57	9.00
	✓	✓	1.92	**2.56**	4.72	5.28		✓	✓	1.70	4.58	6.98	8.67
✓	✓	✓	1.92	**2.56**	4.63	5.21	✓	✓	✓	1.70	4.56	**6.73**	**8.39**

Table 2.2: **Contributions of each energy:** As foreground objects sometimes are texture-less and have large displacement, simple photometric term is not enough. In contrast, background is full of disriminative cues. Simple photometric error would suffice. Adding extra terms will introduce noises and hurt the performance. Please refer to the supp. material for full table.

2.4 Experiments

In this section we first describe the experimental setup. Next we evaluate our model based on pixel-level scene flow metric and instance-level rigid motion metric. Finally we comprehensively study the characteristic of our model.

2.4.1 Dataset and Implementation Details

Data: We validate our approach on the KITTI scene flow dataset [243]. The dataset consists of 200 sets of training images and 200 sets of test images, captured on real world driving scenarios. Following [53], we divide the training data into *train*, *val* splits based on the 4:1 ratio.

Implementation details: For foreground objects, we use all energy terms. The weights are set to 1. For background, we only use photometric term (see ablation study). We run RANSAC 5 times and use the one with lowest mean energy as initialization. We unroll the GN solver for 50 steps. The solver terminates early if the energy reaches plateau. In practice, best energy are often reached within 10 iterations.

2.4.2 Scene Flow Estimation

Comparison to the state-of-the-art: We compare our approach against the leading methods on the benchmark*: ISF [27], PRSM [359], OSF+TC [256], SSF

*As the validation performance of our PWC-Net (fine-tuned on 160 images) performs slightly worse than the official one (fine-tuned on all 200 images), we use their weights instead when submitting

Methods	D1-all	D2-all	Fl-all	SF-all
PSM + PWC	1.89	(47.0)	11.0	(50.8)
Deep+RANSAC	1.89	**2.75**	7.65	8.26
Our Full DRISF	1.89	2.89	**4.10**	**4.84**

Table 2.3: **Improvement over original flow/stereo estimation on validation set:** The numbers in parentheses are obtained by simply warping the disparity output with optical flow, without interpolation, occlusion handling, etc.

Module	Stereo	Optical Flow	Segmentation
Inference time	409 ms / pair	30 ms / pair	251 ms / pair
Module	RANSAC	GN Solver	Total
Inference time	93 ms / instance	244 ms / instance	746 ms / pair

Table 2.4: **Runtime analysis.** Modules within each building block can be executed in parallel (see text for more details).

[295], OSF [243], and CSF [228]. Note that in addition to the standard two adjacent frames, PRSM and OSF+TC rely on extra temporal frames. As shown in Tab. 2.1, our approach (DRISF) outperforms all previous methods by a significant margin in both runtime and outliers ratio. It achieves state-of-the-art performance on almost every entry. DRISF reduces the D1 outliers ratio by **43%**, the D2 outliers ratio by **32%**, and the flow outliers ratio by **24%**. Comparing to ISF model [27], our scene flow error is **22%** lower and our runtime is **800** times faster. Fig. 2-1 compares the performance and runtime of all methods.

Qualitative results: To better understand the pros and cons of our approach, we visualize a few scene flow results on test set in Fig. 2-4. Scene flow estimation is challenging in these scenarios due to large vehicle motions, texture-less regions, occlusion, and illumination variation. For the leftmost image, prior methods fail to estimate the vehicle's motion and adjacent area due to the sun reflection and occlusion. The saturated, high intensity pixels hinder photometric based approaches [243] from matching accurately. With the help of detection and segmentation, ISF [27] is able to improve the foreground estimation. Yet it still fails at the occluded background.

to the benchmark. All other settings remain intact. We thank Deqing Sun for his help.

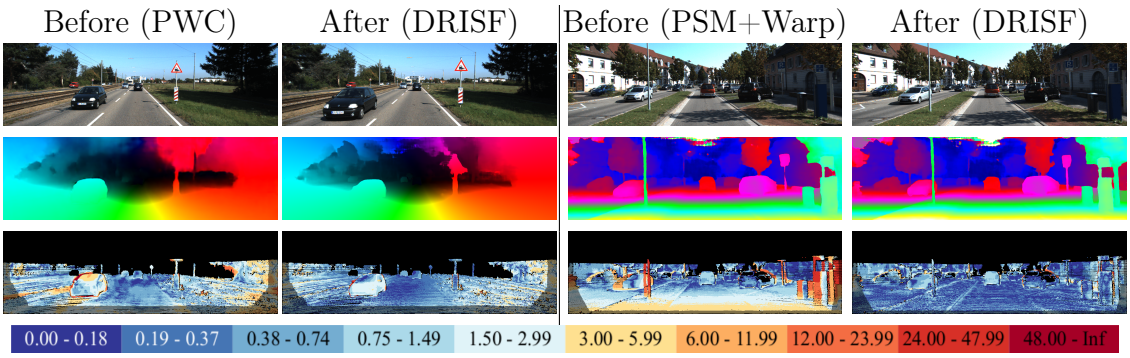


Figure 2-7: **Improvement over original flow/stereo:** DRISF improves the overall performance. It is especially effective on texture-less regions (*e.g.* window of the black car on the left) and occluded areas (right).

In comparison, our approach is robust to illumination changes and is able to handle the occlusion by effectively separating the vehicle from the background. It can also accurately estimate the motion of the small car far away, as well as those of the traffic sticks aside. As we only train our Mask R-CNN on vehicles, it fails to segment the train and hence the failure of our model. For the middle image, the texture-less car has a large displacement and is occluded in the second frame. While previous approaches failed substantially, our method is able to produce accurate motion estimation through the inferred flow and disparity of the remaining non-occluded part. The middle failure mode is again due to the inaccurate segmentation.

2.4.3 3D Rigid Motion Estimation

We now evaluate how good our DRISF model is at estimating the 3D rigid motion. Towards this goal, we exploit the ground truth optical flow, disparity, and instance segmentation provided in the KITTI scene flow dataset to fit a least square rigid motion for each object instance in order to create the ground truth rigid motion.

Curating KITTI scene flow: During fitting, we discover two critical issues with KITTI: first, there are mis-alignments between GT flow/disparity and GT segmentation. Second, the scale fitting of the same 3D CAD model employed to compute ground truth changes sometimes across frames. The first issue is due to the

fact that the GT are collected via different means and thus not consistent. While the GT flow and GT disparity are obtained from the fitted 3D CAD models, the GT segmentation are based on human annotation. To address this, we first use the GT segmentation mask to define each object instance. We then fit a rigid motion using the GT flow and GT disparity of each instance via least squares. Since some boundary pixels may be mis-labeled by the annotators, for each pixel around the boundary we search if there are other instances in the surrounding area, and if there are, we transform the pixel with their rigid motion. If their rigid motion better explains the pixel's 3D movement, *i.e.* the 3D distance is closer, then we assign the pixel to that instance. At the end, we perform the least square fitting again with the new pixel assignment. Unfortunately, even after re-labeling, there are still a few vehicle instances where the rigid motion cannot be explained. After careful diagnose, we notice that this is because the scale of the CAD model changes across frames. To verify our hypothesis, we compute the eigen decomposition for the same instance across frames. Ideally if the scale of the instance does not change much, the eigen value should be roughly the same. Yet we discover a few examples where the largest eigen value changes by 7%. We simply prune those instances as the GT is not accurate.

3D Motion evaluation: Most scene flow methods are pixel-based or adopted a piece-wise rigid setting. It is unclear how to aggregate their estimation into instance-based motion model without affecting their performance. In light of this, we exploit the motion initialization of our GN Solver as baseline. We take the output of the deep nets and apply RANSAC to find the best rigid motion. We denote it as Deep+RANSAC. As shown in Tab. 2.3, this baseline is very competitive. Its performance is comparable to, or even better than prior state-of-the-art. We evaluate our motion model based on translation error and angular error. As shown in Fig. 2-5, over 80% of the vehicles have translation error less than $1m$ and angular error less than $1.3°$. Furthermore, most vehicles with translation error larger than $1m$ is at least $20m$ away. In general, both error slightly increase with distance. This is expected as the farther the vehicle is, the less observations we have. The translation error and angular error are also

strongly correlated.

Visual odometry: The odometry of the 'ego-car' can be computed by estimating the background movement. As a proof-of-concept, we compute the per frame odometry error on the validation images. On average our motion model drifts $0.09m$ and $0.24°$ every $10m$. Fig. 2-6 shows the detailed odometry error w.r.t. the travel distance. We note that the current result is without any pose filter, loop closure, etc. We plan to exploit this direction further in the future.

2.4.4 Analysis

Ablation study: To understand the effectiveness of each energy term on background and foreground objects, we evaluate our model with different energy combinations. As shown in Tab. 2.2, best performance is achieved for foreground objects when using all energy terms, while for background the error is lowest when employing only photometric term. This can be explained by the fact that vehicles are often texture-less, and sometimes have large displacements. If we only employ photometric term, it will be very difficult to establish correspondences and handle drastic appearances changes. With the help of flow and rigid term, we can guide the motion and reduce such effect, and deal with occlusions. In contrast, background is full of discriminative textures and has relatively small motion, which is ideal for photometric term. Adding other terms may introduce extra noise and degrade the performance.

Comparison against original flow/disparity: Through exploiting the structure between visual cues and occlusion handling, our model is able to improve the performance both quantitatively (Tab. 2.3) and qualitatively (Fig. 2-7). The object motion estimation is better, the boundaries are sharper, and the occlusion error is greatly reduced, suggesting that incorporating prior knowledge, such as pixels of same instance should have same rigid motion, into the model is crucial for the task.

Potential improvement To understand the potential gain we may enjoy when improving each module, we sequentially replace the input to our solver with ground truth, one by one, and evaluate our model. Replacing D1 and flow with GT reduce the scene flow error rate by 8% and 21% respectively, while substituting GT for segmentation does not improve the results. This suggests that there are still space for flow and stereo modules to improve.

Runtime analysis We benchmark the runtime of each component in the model during inference in Tab. 2.4. The whole inference pipeline can be decomposed into three sequential stages: visual cues extraction, occlusion reasoning, and optimization. As modules within the same stage are independent, they can be executed in parallel. Furthermore, modern self-driving vehicles are equipped with multiple GPUs. The runtime for each stage is thus the max over all parallel modules. In practice, we exploit two Nvidia 1080Ti GPUs to extract the visual cues: one for PSM-Net, and the other for Mask R-CNN and PWC-Net. Currently, the stereo module takes more than 50% of the overall time. This is largely due to the 3D CNN cost aggregation and the stacked hourglass refinement. In the future, we plan to investigate other faster yet reliable stereo networks. The runtime of the GN solver depends highly on the number of steps we unroll and the number of points we consider. Please refer to the supp. material for detailed analysis.

Limitations: DRISF has two main limitations: first, it heavily depends on the performance of the segmentation network. If the segmentation module fails to detect a vehicle, the vehicle will be treated as background and assigned an inverse ego-car motion. In this case, the 3D motion might be completely wrong, even if the optical flow network accurately predicts its flow. In the future we plan to address this by jointly reasoning about instance segmentation and scene flow. Second, the current energy functions are highly flow centric. Only the photometric term is independent of flow. If the optical flow network completely failed, it would be difficult for the solver to recover the correct motion. One possible solution is thus adding more flow-invariant

energy terms, such as instance association between adjacent frames.

2.5 Conclusion

In this chapter we develop a novel deep structured model for 3D scene flow estimation. We focus on the self-driving scenario where the motion of the scene can be composed by estimating the 3D rigid motion of each actor. We first exploit deep learning to extract visual cues for each instance. Then we employ multiple geometry based energy functions to encode the structural geometric relationship between them. Through optimizing the energy function, we can reason the 3D motion of each traffic participant, and thus scene flow. All operations, including the Gassian-Newton solver, are done in GPU. Our method acheives state-of-the-art performance on the KITTI scene flow dataset. It outperforms all previous methods by a huge margin in both runtime and accuracy. Comparing to prior art, DRISF is 22% better while being two to three orders of magnitude faster.

Chapter 3

Exploiting Semantics Cues and Physical Priors for Robot Localization

Exploiting Sparse Semantic HD Maps for Self-driving Vehicle Localization

Wei-Chiu Ma*, Ignacio Tartavull*, Ioan Andrei Bârsan*, Shenlong Wang*, Min Bai, Gellert Mattyus, Namdar Homayounfar, Shrinidhi Kowshika Lakshmikanth, Andrei Pokrovsky, Raquel Urtasun;

IROS 2019.

Besides the aforementioned geometric structures, physical structures and semantic cues also play an important part in 3D understanding. They help us further make sense of what we see and reason about the unseen. To equip machines with similar capabilities, in this chapter, we develop learning-based models that can exploit physical and semantic priors to ease the learning procedure and produce physically-compliant results. We focus on the task of robot localization where the goal is to infer the pose of the autonomous system within a map. We show that by incorporating the knowledge of the world into the reasoning pipeline we can significantly reduces the difficulty of the problem and build a extremely lightweight yet highly-accurate localization system.

3.1 Introduction

High-definition maps (HD maps) are a fundamental component of most self-driving cars, as they contain useful information about the static part of the environment. The locations of lanes, traffic lights, cross-walks, as well as the associated traffic rules are typically encoded in the maps. They encode the prior knowledge about any scene the autonomous vehicle may encounter.

In order to be able to exploit HD maps, self-driving cars have to localize themselves with respect to the map. The accuracy requirements in localization are very strict and only a few centimeters of error are tolerable in such safety-critical scenarios. Over the past few decades, a wide range of localization systems has been developed. The Global Positioning System (GPS) exploits triangulation from different satellites to determine a receiver's position. It is typically affordable, but often has several meters of error, particularly in the presence of skyscrapers and tunnels. The inertial measurement unit (IMU) computes the vehicle's acceleration, angular rate as well as magnetic field and provides an estimate of its relative motion, but is subject to drift over time.

To overcome the limitations of GPS and IMU, place recognition techniques have been developed. These approaches store what the world looks like either in terms of geometry (*e.g.*, LiDAR point clouds), visual appearance (*e.g.*, SIFT features, LiDAR intensity), and/or semantics (*e.g.*, semantic point cloud), and formulate localization as a retrieval task. Extensions of classical methods such as iterative closest point (ICP) are typically employed for geometry-based localization [410, 5]. Unfortunately, geometric approaches suffer in the presence of repetitive patterns that arise frequently in scenarios such as highways, tunnels, and bridges. Visual recognition approaches [68] pre-record the scene and encode the "landmark" visual features. They then perform localization by matching perceived landmarks to stored ones. However, they often require capturing the same environment for multiple seasons and/or times of the day. Recent work [315] builds dense semantic maps of the environment and combines both semantics and geometry to conduct localization. However, this method requires a large amount of dense map storage and cannot achieve centimeter-level accuracy.

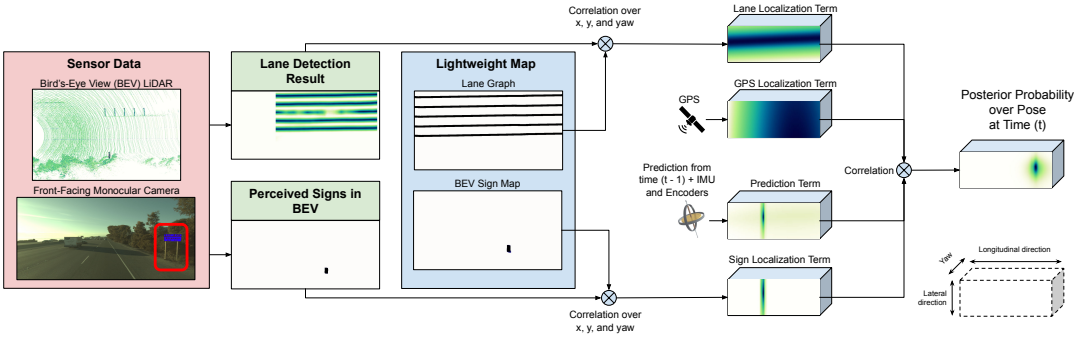


Figure 3-1: **System architecture.** Given the camera image and LiDAR sweep as input, we first detect lanes in the form of truncated inverse distance transform and detect signs as a bird's-eye view (BEV) probability map. The detection output is then passed through a differentiable rigid transform layer [149] under multiple rotational angles. Finally, the inner-product score is measured between the inferred semantics and the map. The probability score is merged with GPS and vehicle dynamics observations and the inferred pose is computed from the posterior using soft-argmax. The camera image on the left contains an example of a sign used in localization, highlighted with the red box.

While place recognition approaches are typically fairly accurate, the costs associated with ensuring the stored representations are up to date can often be prohibitive. They also require very large storage on board. Several approaches have been proposed to provide affordable solutions to localization by exploiting coarse maps that are freely available on the web [37, 229]. Despite demonstrating promising results, the accuracy of such methods is still in the order of a few meters, which does not meet the requirements of safety-critical applications such as autonomous driving.

With these challenges in mind, in this chapter we propose a lightweight localization method that does not require detailed knowledge about the appearance of the world (*e.g.*, dense geometry or texture). Instead, we exploit vehicle dynamics as well as a semantic map containing lane graphs and the locations of traffic signs. Traffic signs provide information in longitudinal direction, while lanes help avoid lateral drift. These cues are complementary to each other and the resulting maps can be stored in a fraction of the memory necessary for traditional HD maps, which is important as self-driving cars need to operate in very large environments. We formulate the localization problem as a Bayes filter, and demonstrate the effectiveness of our approach on North-American highways, which are challenging for current place recognition approaches

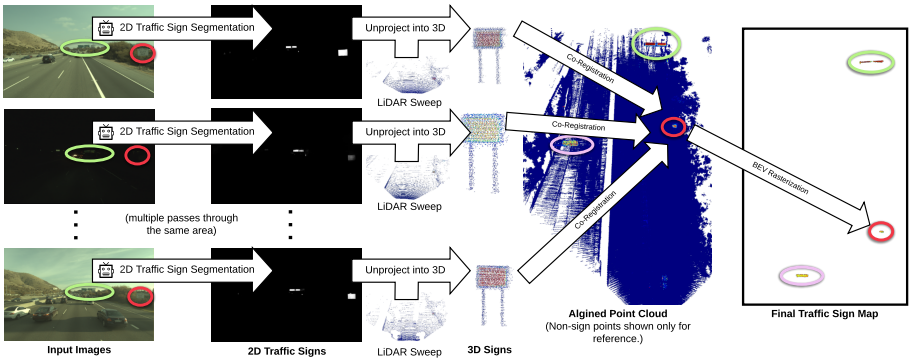


Figure 3-2: **Traffic sign map building process.** We first detect signs in 2D using semantic segmentation in the camera frame, and then use the LiDAR points to localize the signs in 3D. Mapping can aggregate information from multiple passes through the same area using the ground truth pose information, and can function even in low light, as highlighted in the middle row, where the signs are correctly segmented even at night time. This information is used to build the traffic sign map in a fully automated way.

as repetitive patterns are common and driving speeds are high. Our experiments on more than 300 km of testing trips showcase that we are able to achieve 0.05m median lateral accuracy and 1.12m median longitudinal accuracy, while using roughly three orders of magnitude less storage than previous map-based approaches (0.55MiB/km^2 vs. the 1.4GiB/km^2 required for dense point clouds).

3.2 Related Work

Place recognition: One of the most prevailing approaches in self-localization is place recognition [13, 18, 125, 311, 197, 247, 386, 12, 414]. By recording the appearance of the world and building a database of it in advance, the localization task can be formulated as a retrieval problem. At test time, the system simply searches for the most similar scene and retrieves its pose. As most of the features used to describe the scene (e.g., 3D line segments [18] or 3D point clouds [197, 247, 386]), are highly correlated with the appearance of the world, one needs to update the database frequently. With this problem in mind, [68, 255, 215] proposed an image-based localization technique that is to some degree invariant to appearance changes. More recently, [169] designed a CNN to directly estimate the pose of the camera. While their method is robust

to illumination changes and weather, they still require training data for each scene, limiting its scalability.

Geometry-based localization: Perspective-n-Point (PnP) approaches have been used for localization. The idea is to extract local features from images and find correspondences with the pre-stored geo-registered point sets. For instance, [321] utilized random forests to find correspondence between RGBD images and pre-constructed 3D indoor geometry. Li et al. [197] pre-stored point clouds along with SIFT features for this task, while Liu et al. [219] proposed to use branch-and-bound to solve the exact 2D-3D registration. However, these approaches require computing a 3D reconstruction of the scene in advance, and do not work well in scenarios with repetitive geometric structures.

Simultaneous localization and mapping: Given a sequence of images, point clouds, or depth images, SLAM approaches [90, 250, 420] estimate relative camera poses between consecutive frames through feature matching and joint pose estimation. Accumulated errors make the localization gradually drift as the robot moves. In indoor or urban scenes, loop closure has been used to fix the accumulated errors. However, unlike indoor or urban scenarios, on highways trajectories are unlikely to be closed, which makes drift a much more challenging problem to overcome.

Lightweight localization: There is a growing interest in developing affordable localization techniques. Given an initial estimate of the vehicle position, [99] exploited ego-trajectory to self-localize within a small region. Brubaker et al. [37] developed a technique that can be applied at city scale, without any prior knowledge about the vehicle location. Ma et al. [229] incorporated other visual cues, such as the position of the sun and the road type to further improve the results. These works are appealing since they only require a cartographic map. However, the localization accuracy is strongly limited by the performance of odometry. The semantic cues are only used to resolve ambiguous modes and speed up the inference procedure. Second, the computational complexity is a function of the uncertainty in the map, which remains fairly large when dealing with maps that have repetitive structures.

High-precision Map-based Localization: The proposed work belongs to the

category of the high-precision map-based localization [192, 191, 387, 386, 317, 410, 440, 23]. The use of maps has been shown to not only provide strong cues for various tasks in computer vision and robotics such as scene understanding [366], vehicle detection [241], and localization [37, 367, 229], but also enables the creation of large-scale datasets with little human effort [380, 371]. The general idea is to build a centimeter-level high-definition 3D map offline a priori, by stitching sensor input using a high-precision differential GNSS system and offline SLAM. Early approaches utilize LiDAR sensors to build maps [192]. Uncertainty in intensity changes have been handled through building probabilistic prior map [191, 387]. In the online stage, the position is determined by matching the sensor reading to the prior map. For instance, [192, 191, 387] utilized the perceived LiDAR intensity to conduct matching. Yoneda et al. [410] proposed to align online LiDAR sweeps against an existing 3D prior map using ICP, [440, 386] utilized visual cues from cameras to localize the self-driving vehicles, and [23] use a fully convolutional neural network to learn the task of online-to-map matching in order to improve robustness to dynamic objects and eliminate the need for LiDAR intensity calibration.

Semantic localization: Schreiber et al. [317] proposed to use lanes as localization cues. Towards this goal, they manually annotated lane markings over the LiDAR intensity map. The lane markings are then detected online using a stereo camera, and matched against the ones in the map. Welzel et al. [383] and Qu et al. [282] utilize traffic signs to assist image-based vehicle localization. Specifically, traffic signs are detected from images and matched against a geo-referenced sign database, after which local bundle adjustment is conducted to estimate a fine-grained pose. More recently, [315] built dense semantic maps using image segmentation and conducted localization by matching both semantic and geometric cues. In contrast, the maps used in our approach only need to contain the lane graphs and the inferred sign map, the latter of which is computed without a human in the loop, while also only requiring a fraction of the storage used by dense maps.

3.3 Lightweight HD Mapping

In order to conduct efficient and accurate localization, a compressed yet informative representation of the world needs to be constructed. Ideally our HD maps should be easy to (automatically) build and maintain at scale, while also enabling real-time broadcasting of map changes between a central server and the fleet. This places stringent storage requirements that traditional dense HD maps fail to satisfy. In this chapter, we tackle these challenges by building sparse HD maps containing just the lane graph and the locations of traffic signs. These modalities provide complementary semantic cues for localization. Importantly, the storage needs for our maps are three orders of magnitude smaller than traditional LiDAR intensity maps [191, 387, 192, 23] or geometric maps [410].

Lane Graph Most roads have visually distinctive lanes determining the expected trajectory of vehicles, compliant with the traffic rules. Most self driving cars store this prior knowledge as lane graphs $\mathcal{L}$. A lane graph is a structured representation of the road network defined as a set of polygonal chains (polylines), each of which represents a lane boundary. We refer the reader to Fig. 3-1 for an illustration of a lane graph. Lane graphs provide useful cues for localization, particularly in the lateral position and the heading of the vehicle.

Traffic Signs Traffic signs are common semantic landmarks that are sparsely yet systematically present in cities, rural areas, and highways. Their presence provides useful cues that can be employed for accurate longitudinal localization. In this chapter we build sparse HD maps containing traffic signs automatically. Towards this goal, we exploit multiple passes of our vehicles over the same region and identify the signs by exploiting image-based semantic segmentation followed by 3D sign localization using LiDAR via inverse-projection from pixel to 3D space. A consistent coordinate system over the multiple passes is obtained by means of offline multi-sensor SLAM. Note that in our map we only store points that are estimated to be traffic signs above a certain confidence level. After that, we rasterize the sparse points to create the traffic sign

presence probability map $\mathcal{T}$ in bird's-eye view (BEV) at 5cm per pixel. This is a very sparse representation containing all the traffic signs. The full process is conducted without any human intervention. Fig. 3-2 depicts the traffic sign map building process and an example of its output.

3.4 Localization as Bayes Inference with Deep Semantics

In this chapter we propose a novel localization system that exploits vehicle dynamics as well as a semantic map containing both a lane graph and the locations of traffic signs. These cues are complementary to each other and the resulting maps can be stored in a fraction of the memory necessary for traditional dense HD maps. We formulate the localization problem as a histogram filter taking as input the structured outputs of our sign and lane detection neural networks, as well as GPS, IMU, and wheel odometry information, and outputting a probability histogram over the vehicle's pose, expressed in world coordinates.

3.4.1 Probabilistic Pose Filter Formulation

Our localization system exploits a wide variety of sensors: GPS, IMU, wheel encoders, LiDAR, and cameras. These sensors are available in most self-driving vehicles. The GPS provides a coarse location with several meters accuracy; an IMU captures vehicle dynamic measurements; the wheel encoders measure the total travel distance; the LiDAR accurately perceives the geometry of the surrounding area through a sparse point cloud; images capture dense and rich appearance information. We assume our sensors are calibrated and neglect the effects of suspension, unbalanced tires, and vibration. As shown in our experiments, the influence of these factors is negligible and other aspects such as sloped roads (*e.g.*, on highway ramps) do not have an impact on our localizer. Therefore, the vehicle's pose can be parametrized with only three degrees of freedom (instead of six) consisting of a 2D translation and a heading angle

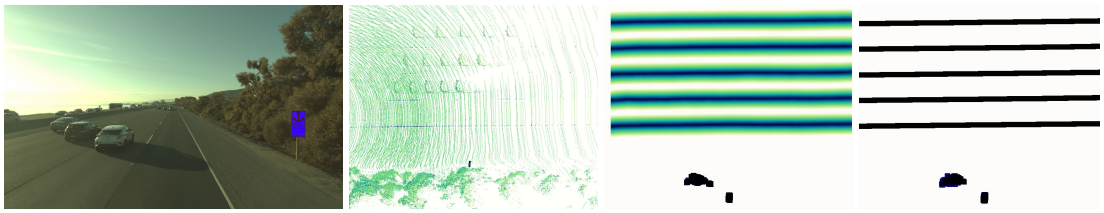


Figure 3-3: **Dataset sample and inference results.** Our system detects signs in the camera images (note the blue rectangle on the right side of the first image) and projects the sign's points in a top-down view using LiDAR (second image). It uses this result in conjunction with the lane detection result (third image) to localize against a lightweight map consisting of just signs and lane boundaries (fourth image).

w.r.t. the map coordinate's origin, *i.e.* $\mathbf{x} = \{\mathbf{t}, \theta\}$, where $\mathbf{t} \in \mathbb{R}^2$ and $\theta \in (-\pi, \pi]$, since the heading is parallel to the ground plane.

Following similar concepts to [23], we factorize the posterior distribution over the vehicle's pose into components corresponding to each modality, as shown in Eq. (3.1).

Let $\mathcal{G}_t$ be the GPS readings at time t and let $\mathcal{L}$ and $\mathcal{T}$ represent the lane graph and traffic sign maps respectively. We compute an estimate of the vehicle dynamics $\mathcal{X}_t$ from both IMU and the wheel encoders smoothed through an extended Kalman filter, which is updated at 100Hz.

The localization task is formulated as a histogram filter aiming to maximize the agreement between the observed and mapped lane graphs and traffic signs while respecting vehicle dynamics:

$$\begin{aligned} \text{Bel}_t(\mathbf{x}) = \eta \cdot & P_{\text{LANE}}(\mathcal{S}_t|\mathbf{x}, \mathcal{L}; \mathbf{w}_{\text{LANE}}) P_{\text{SIGN}}(\mathcal{S}_t|\mathbf{x}, \mathcal{T}; \mathbf{w}_{\text{SIGN}}) \\ & P_{\text{GPS}}(\mathcal{G}_t|\mathbf{x}) \text{Bel}_{t|t-1}(\mathbf{x}|\mathcal{X}_t), \end{aligned} \tag{3.1}$$

where $\text{Bel}_t(\mathbf{x})$ is the posterior probability of the vehicle pose at time t; η is a normalizing factor to ensure sum of all probability is equal to one; $\mathbf{w}_{\text{LANE}}$ and $\mathbf{w}_{\text{SIGN}}$ are sets of learnable parameters, and $\mathcal{S}_t = (\mathcal{I}_t, \mathcal{C}_t)$ is a sensory measurement tuple composed from LiDAR $\mathcal{I}_t$ and camera $\mathcal{C}_t$. Note that by recursively solving Eq. 3.1, we can localize the vehicle at every step with an uncertainty measure that could be propagated to the next step. We now describe each energy term and our inference algorithm in more details.

Lane Observation Model We define our matching energy to encode the agreement between the lane observation from the sensory input and the map. Our probability is computed by a normalized matching score function that utilizes the existing lane graph and compares it to detected lanes. To detect lanes we exploit a state-of-the-art real-time multi-sensor convolutional network [16]. The input of the network is a front-view camera image and raw LiDAR intensity measurement projected onto BEV. The output of the network is the inverse truncated distance function to the lane graph in the overhead view. Specifically, each pixel in the overhead view encodes the Euclidean distance to the closest lane marking, up to a truncation threshold of 1m. We refer the reader to Fig. 3-3 for an illustration of the neural network's input and output.

To compute the probability, we first orthographically project the lane graph $\mathcal{L}$ onto overhead view such that the lane detection output and the map are under the same coordinate system. The overhead view of the lane graph is also represented using an truncated inverse distance function. Given a vehicle pose hypothesis $\mathbf{x}$, we rotate and translate the lane detection prediction accordingly and compute its matching score against the lane graph map. The matching score is an inner product between the lane detection and the lane graph map

$$P_{\text{LANE}} \propto s\left(\pi\left(f_{\text{LANE}}(\mathcal{S}; \mathbf{w}_{\text{LANE}}), \mathbf{x}\right), \mathcal{L}\right), \tag{3.2}$$

where f_{LANE} is the deep lane detection network and $\mathbf{w}_{\text{LANE}}$ are the network's parameters. π is a 2D rigid transform function to transform the online lane detection to the map's coordinate system given a pose hypothesis $\mathbf{x}$; $s(\cdot, \cdot)$ is a cross-correlation operation between two images.

Traffic Sign Observation Model This model encodes the consistency between perceived online traffic signs and the map. Specifically, we run an image-based semantic segmentation algorithm that performs dense semantic labeling of traffic signs. We adopt the state-of-the-art PSPnet structure [433] to our task. The encoder

architecture is a ResNet50 backbone and the decoder is a pyramid spatial pooling network. Two additional convolutional layers are added in the decoder stage to further boost performance. The model is jointly trained with the instance segmentation loss following [14]. Fig. 3-3 depicts examples of the network's input and output. The estimated image-based traffic sign probabilities are converted onto the overhead view to form our online traffic sign probability map. This is achieved by associating each LiDAR with a pixel in the image by projection. We then read the softmax probability of the pixel's segmentation as our estimate. Only high-confident traffic sign pixels are unprojected to 3D and rasterized in BEV. Given a pose proposal $\mathbf{x}$, we define the sign matching probability analogously to the lane matching one as

$$P_{\text{SIGN}} \propto s\left(\pi\left(f_{\text{SIGN}}(\mathcal{S}; \mathbf{w}_{\text{SIGN}}), \mathbf{x}\right), \mathcal{T}\right), \tag{3.3}$$

where f_{SIGN} is the sign segmentation network and $\mathbf{w}_{\text{SIGN}}$ are the networks' parameters. Both the perceived signs, as well as the map they are matched against are encoded as pixel-wise occupancy probabilities.

GPS Observation Model This term encodes the likelihood of the GPS sensory observation $\mathcal{G}$ at a given vehicle pose $\mathbf{x}$:

$$P_{\text{GPS}} \propto \exp\left(-\frac{(g_x - x)^2 + (g_y - y)^2}{\sigma_{\text{GPS}}^2}\right), \tag{3.4}$$

where $[g_x, g_y]^T = T \cdot \mathcal{G}$ represents a GPS point location in the coordinate frame of the map against which we are localizing. T is the given rigid transform between the Universal Transverse Mercator (UTM) coordinates and the map coordinates and $\mathcal{G}$ is the GPS observation expressed in UTM coordinates.

Dynamics Model This term encourages consistency between the pose proposal $\mathbf{x}$ and the vehicle dynamics estimation, given the previous vehicle pose distribution. The pose at the current timestamp depends on previous pose and the vehicle motion. Given an observation of the vehicle motion $\mathcal{X}_t$, the motion model is computed by

marginalizing out the previous pose:

$$\text{Bel}_{t|t-1}(\mathbf{x}|\mathcal{X}_t) = \sum_{\mathbf{x}_{t-1}\in\mathcal{R}_{t-1}} P(\mathbf{x}|\mathcal{X}_t, \mathbf{x}_{t-1})\text{Bel}_{t-1}(\mathbf{x}_{t-1}) \tag{3.5}$$

where the likelihood is a Gaussian probability model

$$P(\mathbf{x}|\mathcal{X}_t, \mathbf{x}_{t-1}) \propto \mathcal{N}((\mathbf{x} \ominus (\mathbf{x}_{t-1} \oplus \mathcal{X}_t)), \Sigma) \tag{3.6}$$

with Σ the covariance matrix and $\mathcal{R}_{t-1}$ is the search space for $\mathbf{x}_{t-1}$. In practice, we only need $\mathcal{R}_{t-1}$ to be a small local region centered at $\mathbf{x}^*_{t-1}$ given the fact the rest of the pose space has negligible probability. Note that $\oplus$, $\ominus$ are the standard 2D pose composition and inverse pose composition operators described by Kummerle et al. [180].

3.4.2 Efficient Inference

Discretization The inference defined in Eq. (3.1) is intractable. Following [23, 191] we tackle this problem using a histogram filter. We discretize the full continuous search space over $\mathbf{x} = \{x, y, \theta\}$ into a search grid, each with associated posterior Bel($\mathbf{x}$). We restrict the search space to a small local region at each time. This is a reasonable assumption given the constraints of the vehicles dynamics at a limited time interval.

Accelerating Correlation We now discuss the computation required for each term. We utilize efficient convolution-based exhaustive search to compute the lane and traffic sign probability model. In particular, enumerating the full translational search range with inner product is equivalent to a correlation filter with a large kernel (which is the online sign/lane observation). Motivated by the fact that the kernel is very large, FFT-conv is used to accelerate the computation speed by a factor of 20 over the state-of-the-art GEMM-based spatial GPU correlation implementations [23].

Robust Point Estimation Unlike the MAP-inference which simply takes the configuration which maximizes the posterior belief, we adopt a center-of-mass based

soft-argmax [191] to better incorporate the uncertainty of our model and encourage smoothness in our localization. We thus define

$$\mathbf{x}_t^* = \frac{\sum_{\mathbf{x}} \text{Bel}_t(\mathbf{x})^\alpha \cdot \mathbf{x}}{\sum_{\mathbf{x}} \text{Bel}_t(\mathbf{x})^\alpha}, \tag{3.7}$$

where $\alpha \geq 1$ is a temperature hyper-parameter. This gives us an estimate that takes the uncertainty of the prediction into account.

3.4.3 Learning

Both the lane detection network and the traffic sign segmentation network are trained through back-propagation separately using ground-truth annotated data. The lane detection is trained with a regression loss that measures the ℓ_2 distance between the predicted inverse truncated distance transform and the ground-truth one [16]. The semantic segmentation network is trained with cross-entropy [14]. Hyper-parameters for the Bayes filter (*e.g.*, σ^2_{GPS}, softmax temperature α, etc.) are searched through cross-validation.

3.5 Experiments

We validate the effectiveness of our localization system on a highway dataset of 312km. We evaluate our model in terms of its localization accuracy and runtime.

3.5.1 Dataset

Our goal is to perform fine-grained localization on highways. Unfortunately, there is no publicly available dataset that provides ground truth localization at the centimeter-level precision required for safe autonomous driving. We therefore collected a dataset of highways by driving over 300km in North America at different times of the year, covering over 100km of roads. The dataset encompasses 64-line LiDAR sweeps and images from a front-facing global shutter camera with a resolution of 1900×1280, both captured at 10Hz, as well as IMU and GPS sensory data and the lane graphs.

The extrinsic calibration between the camera and LiDAR is conducted using a set of calibration targets [122]. The ground truth 3D localization is estimated by a high-precision ICP-based offline Graph-SLAM using high-definition pre-scanned scene geometry. Fig. 3-3 shows a sample from our dataset together with the inferred and ground truth lane graphs.

Our dataset is partitioned into 'snippets', each consisting of roughly 2km of driving. The training, validation, and test splits are conducted at the snippet level, where training snippets are used for map building and training the lane detection network, and validation snippets are used for hyper-parameter tuning. The test snippets are used to compute the final metrics. An additional 5,000 images have been annotated with pixel-wise traffic sign labels which are used for training the sign segmentation network.

3.5.2 Implementation Details

Network training: To train the lane detection network, we uniformly sample 50K frames from the training region based on their geographic coordinates to train the network. The ground truth can be automatically generated given the vehicle pose and the lane graph. We use a mini-batch size of 16 and employ Adam [173] as the optimizer. We set the learning rate to 10^{-4}. The network was trained completely from scratch with Gaussian initialization and converged roughly after 10 epochs. We visualize some results in Fig. 3-3.

We train our traffic sign segmentation network separately over four GPUs with a total mini-batch size of 8. Synchronized batch normalization is utilized for multi-GPU batch normalization. The learning rate is set to be 10^{-4} and the network is trained from scratch. The backbone of the model is fine-tuned from a DeepLab v2 network pre-trained over the Pascal VOC dataset.

Hyper-parameter search: We choose the hyper-parameters through grid search over a mini-validation dataset consists of 20 snippets of 2km driving. The hyper-parameters include the temperatures of the final pose soft-argmax, the lane probability softmax, and the sign probability softmax, as well as the observation noise parameters

for GPS and the dynamics. The best configuration is chosen by the failure rate metric. In the context of hyperparameter search, the failure rate is a snippet-level metric which counts a test snippet as failed if the total error becomes greater than 1m at any point. We therefore picked the hyperparameter configuration which minimized this metric on our validation set, and kept it fixed at test time. As noted in Sec. 3.4.2, we restrict our search range to a small area centered at the dead reckoning pose and neglect the probability outside the region. We notice in practice that thanks to the consistent presence of the lanes in self-driving scenarios, there is less uncertainty along the lateral direction than along the longitudinal. The presence of traffic signs helps reduce uncertainty along the longitudinal direction, but signs could be as sparse as every 1km, during which INS drift could be as large as 7 meters. Based on this observation and with the potential drift in mind, we choose a very conservative search range $B = B_x \times B_y \times B_\theta = [-0.75m, 0.75m] \times [-7.5m, 7.5m] \times [-2^\circ, 2^\circ]$ at a spatial resolution of 5cm and an angular resolution of 1°.

3.5.3 Localization

Metrics: We adopt several key metrics to measure the localization performance of the algorithms evaluated in this Section.

In order to safely drive from a certain point to another without any human intervention, an autonomous vehicle must be aware of where it is w.r.t. the map. Lateral error and longitudinal error have different meanings for self-driving since a small lateral error could result in localizing in the wrong lane, while ambiguities about the longitudinal position of the vehicle are more tolerable. As localization is the first stage in self-driving pipeline, it it critical that it is robust enough with a very small failure rate; therefore, understanding worst-case performance is critical.

Moreover, localization results should reflect the vehicle dynamics as well, which ensures the smoothness of decision making, since sudden jumps in localization might cause downstream components to fail. To this end, we also measure the prediction smoothness of our methods. We define smoothness as the difference between the temporal gradient of the ground truth pose and that of the predicted poe. We estimate

Methods	Longitudinal Error (m)			Lateral Error (m)		
	Median	95%	99%	Median	95%	99%
Dynamics	24.85	128.21	310.50	114.46	779.33	784.22
GPS	1.16	5.78	6.76	1.25	8.56	9.44
INS	1.59	6.89	13.62	2.34	11.02	42.34
Ours	**1.12**	**3.55**	**5.92**	**0.05**	**0.18**	**0.23**

Table 3.1: **Quantitative results on localization accuracy.** 'Ours' refers to our full model where we employ dynamics, GPS, lanes, and signs, in a probabilistic manner.

the gradients using first-order finite differences, i.e., by simply taking the differences between poses at times (t) and $(t-1)$. As such, we define smoothness as

$$s = \frac{1}{T}\sum_{t=1}^{T} \left\| (\mathbf{x}_t^* - \mathbf{x}_{t-1}^*) - (\mathbf{x}_t^{GT} - \mathbf{x}_{t-1}^{GT}) \right\|^2 . \tag{3.8}$$

Baselines: We compare our results with two baselines: dynamics and dynamics+GPS. The first baseline builds on top of the dynamics of the vehicle. It takes as input the IMU data and wheel odometry, and use the measurements to extrapolate the vehicle's motion. The second baseline employs histogram filters to fuse information between IMU readings and GPS sensory input, which combines motion and absolute position cues.

Quantitative analysis: As shown in Tab. 3.1 and 3.2, our method significantly outperforms the baselines across all metrics. To be more specific, our model has a median longitudinal error of 1.12m and a median lateral error of 0.06m; both are much smaller than other competing methods, with lateral error one order of magnitude lower. We notice that our method greatly improve the performance over the worst case scenario in terms of both longitudinal error, lateral error, and smoothness.

Qualitative results: We show the localization results of our system as well as those of the baselines in Fig. 3-4. Through lane observations, our model is able to consistently achieve centimeter-level lateral localization accuracy. When signs are visible, the traffic sign model helps push the prediction towards the location where the observation and map have agreement, bringing the pose estimate to the correct longitudinal position. In contrast, GPS tends to produce noisy results, but helps

Method Name	Smoothness			
	Mean	95%	99%	Max
Dynamics	0.2	0.4	0.6	1.2
GPS	**0.1**	0.2	0.3	8.5
INS	**0.1**	**0.1**	**0.2**	3.7
Ours	**0.1**	0.2	0.3	**0.9**

Table 3.2: **Quantitative comparison on smoothness.**

Method	Properties			Travelling Dist = $2km$					
				Longitudinal Error (m)			Lateral Error (m)		
	Lane	GPS	Sign	Median	95%	99%	Median	95%	99%
Lane	yes	no	no	13.45	37.86	51.59	0.20	1.08	1.59
Lane+GPS	yes	yes	no	1.53	5.95	6.27	0.06	0.24	0.43
Lane+Sign	yes	no	yes	6.23	31.98	51.70	0.10	0.85	1.41
All	yes	yes	yes	**1.12**	**3.55**	**5.92**	**0.05**	**0.18**	**0.23**

Table 3.3: **Contribution of each component on localization.**

substantially improve worst-case performance.

Runtime analysis: To further demonstrate that our localization system is of practical usage, we benchmark the runtime of each component in the model during inference using an NVIDIA GTX 1080 GPU. A single step of our inference takes 153ms in total on average, with 32ms on lane detection, 110ms on semantic segmentation and 11ms on matching, which is roughly 7 fps. We note that the real-time performance is made possible largely with the help of FFT convolutions.

Map storage analysis: We compare the size of our HD map against other commonly used representations: LiDAR intensity map and 3D point cloud map. For a fair comparison, we store all data in a lossless manner and measure the storage requirements. While the LiDAR intensity and 3D point cloud maps consume 177 MiB/km^2 and 1,447 MiB/km^2 respectively, our HD map only requires **0.55 MiB** per square kilometer. This is only 0.3% of the size of LiDAR intensity map and 0.03% of that of 3D point cloud map.

Ablation study: To better understand the contribution of each component of our model, we respectively compute the longitudinal and lateral error under diverse settings.

Inference	Travelling Dist = $2km$						Smoothness			
	Longitudinal Error (m)			Lateral Error (m)						
	Median	95%	99%	Median	95%	99%	Mean	95%	99%	Max
Deterministic	1.29	3.65	**5.16**	0.08	0.26	0.50	0.11	**0.19**	1.78	5.27
Probabilistic	**1.12**	**3.55**	5.92	**0.05**	**0.18**	**0.23**	**0.07**	**0.19**	**0.24**	**0.98**

Table 3.4: **Probabilistic vs deterministic:** Quantitative comparison of the localization system when using a probabilistic filtering or deterministic model (with full observations: Lane+GPS+Sign).

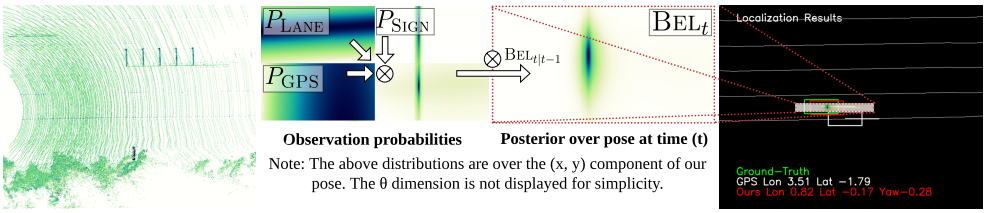


Figure 3-4: **Qualitative results.** A bird's-eye view of the last five LiDAR sweeps (left), which are used for the lane detection, together with the observation probabilities and the posterior (middle), followed by a comparison between the localization result, the ground truth pose, and GPS (right). The (x, y)-resolution of each probability distribution is 1.5m laterally (vertical) and 15m longitudinally (horizontal).

As shown in Tab. 3.3, each term (GPS, lane, sign) has a positive contribution to the localization performance. Specifically, the lane observation model greatly increases lateral accuracy, while sign observations increase longitudinal accuracy. We also compare our probabilistic histogram filter formulation with a deterministic model. Compared against our histogram filter approach, the non-probabilistic one performs a weighted average between each observation without carrying over the previous step's uncertainty. As shown in Tab. 3.4, by combining all the observation models, the non-probabilistic model can achieve reasonable performance but still remains less accurate than the probabilistic formulation. Moreover, due to the fact that no uncertainty history is carried over, prediction smoothness over time is not guaranteed.

Chapter 4

Deep Feedback Inverse Problem Solver

Deep Feedback Inverse Problem Solver
Wei-Chiu Ma, Shenlong Wang, Jiayuan Gu, Sivabalan Manivasagam, Antonio Torralba, Raquel Urtasun;
ECCV 2020.

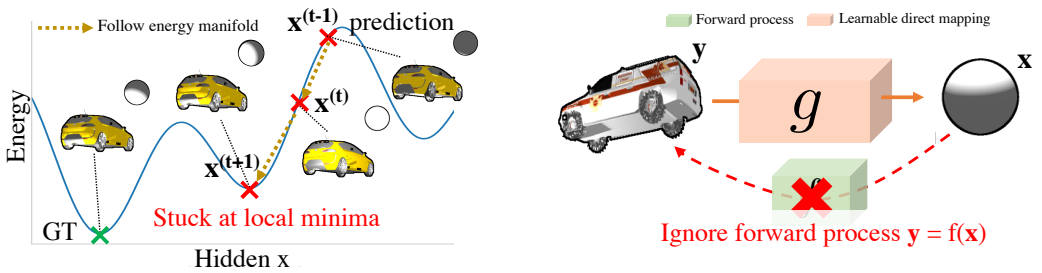
So far we have shown that leveraging geometric and physical insights into model design can prevent the model from learning natural laws from scratch and thus can utilize the data much more efficiently. In this chapter, we further demonstrate that besides easing the learning procedure and guaranteeing physically-compliant results, with proper design, physical processes can also serve as supervisory training signals. Specifically, we develop a generic closed-loop network that leverages the feedback signal provided by the known physical forward process to iteratively solve different inverse graphic tasks. Our approach is fast, effectively, computationally efficient, and can be applied to a wide range of 3D inverse problems.

4.1 Introduction

Given a 3D model of an object, the light source(s), and their relevant poses to the camera, one can generate highly realistic images of the scene with one click. While such a *forward* rendering process is complicated and requires explicit modeling of interreflections, self-occlusions, as well as distortions, it is well-defined and can be computed effectively. However, if we were to recover the illumination parameters or predict the 6 DoF pose of the object from the image in an *inverse* fashion, the task becomes extremely challenging. This is because a lot of crucial information is lost during the forward (rendering) process. In fact, many complicated tasks in natural science, signal processing, and robotics, all face similar challenges – the model parameters of interest cannot be measured directly and need to be estimated from limited observations. This family of problems are commonly referred to as **inverse problems**. Unfortunately, while there exists sophisticated theories on how to design the forward processes, how to address the inherent ambiguities of the inverse problem still remains an open question.

One popular strategy to disambiguate the solution is to model the inverse problem as a structured optimization task and incorporate human knowledge into the model [126, 268, 141, 304]. For instance, the estimated solution should agree with the observations [267] and be smooth [263, 28], or should follow a certain statistical distribution [190, 376, 19]. Through imposing carefully designed objectives, classic structure optimization methods are able to find a solution that not only agrees with the observations but also satisfies our prior knowledge about the solution. In practice, however, almost no hand-crafted priors can succeed in including all phenomena. To ensure that the optimization problem can be solved efficiently, there are multiple restrictions on the form of these priors as well as the the forward process [33], both of which increase the difficulty of design. Furthermore, most optimization approaches require many iterations to converge and are sensitive to initialization.

On the other hand, learning based methods propose to directly learn a mapping from observations to the model parameters [350, 163, 407, 389, 298]. They capitalize

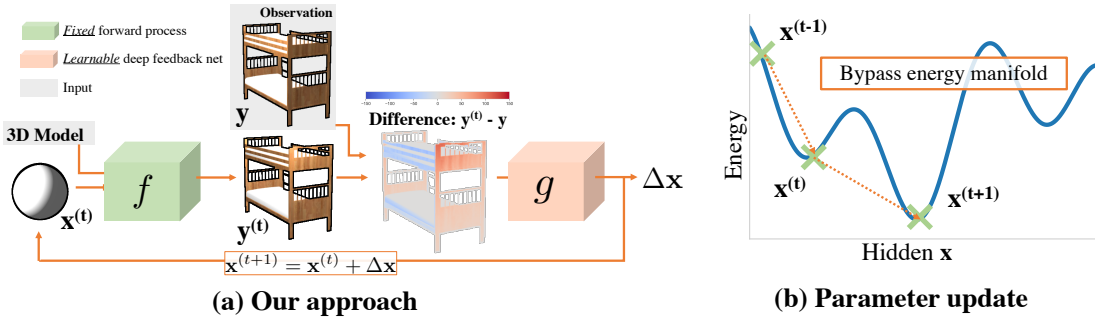


(a) Structure optimization **(b) Learning based methods**

Figure 4-1: **Prior work on inverse problems**: (a) Structured optimization approaches require hand-crafted energy/objective functions and are sensitive to initializations which makes them easy to get stuck in local optima. (b) Direct learning based methods do not utilize the available forward process as feedback to guarantee the quality of the solution. Without this feedback, the models cannot rectify the estimates effectively as shown above.

on powerful machine learning tools to extract task-specific priors in a data-driven fashion. With the help of large-scale datasets and the flourishing of deep learning, they are able to achieve state-of-the-art performance on a variety of inverse problems [390, 351, 320, 91, 162, 230]. Unfortunately, these methods often ignore the fact that the forward model for inverse problems is available. Their systems remain open loop and do not have the capability to *update* their prediction based on the *feedback signal*. Consequently, the estimated parameters, while performing well in the majority of the cases, may generate results that are either incompatible with the real observations or not realistic.

With these challenges in mind, we develop a novel approach to solving inverse problems that takes the best of both worlds. The key idea is to *learn to iteratively update* the current estimation through the *feedback signal* from the forward process. Specifically, we design a neural network that takes the observation and the forward simulation result of the previous estimation as input, and outputs a steep update towards the ideal model parameters. The advantages are four-fold: First, as each update is trained to aggressively move towards the ground truth, we can accelerate the update procedure and reach the target with much fewer iterations than classic optimization approaches. Second, our approach does not need to explicitly define the energy. Third, we do not have any restrictions on the forward process, such as



(a) Our approach **(b) Parameter update**

Figure 4-2: **Overview:** Our model iteratively updates the estimation based on the feedback signal from the forward process. We adopt a closed-loop scheme to ensure the consistency between the estimation and the observation. We neither require an objective at test time, nor have any restrictions on the forward process. Click here to watch an animated version of the update procedure.

differentiability, which greatly expands the applicable domain. Finally, in contrast to conventional learning methods, our method incorporates feedback signals from the forward process so that the network is aware of how close the current estimation is to the ground truth and can react accordingly. The estimated parameters generally lead to results closer to the observation.

We demonstrate the effectiveness of our approach on three different inverse problems in graphics and robotics: illumination estimation, 6 DoF pose estimation, and inverse kinematics. Compared to traditional optimization based methods, we are able to achieve comparable or better performance while being two to three orders of magnitude faster. Compared to deep learning based approaches, our model consistently improves the performance on all metrics.

4.2 Background

Let $\mathbf{x} \in \mathcal{X}$ be the hidden parameters of interest and let $\mathbf{y} \in \mathcal{Y}$ be the measurable observations. Denote $f : \mathbf{x} \to \mathbf{y}$ as the deterministic forward process. The aim of inverse problem is to recover $\mathbf{x}$ given the observation $\mathbf{y}$ and the forward mapping f. In the tasks that we consider, $\mathcal{X}$ is a group such as $\mathcal{X} = \mathrm{SE}(3)$ for 6 DoF pose estimation and $\mathcal{X} = \mathbb{R}^3$ when estimating the position of the light source

4.2.1 Structured optimization

Structured optimization methods generally formulate the inverse problem as an energy minimization task [50, 76, 78, 277, 182, 267, 141]:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} E(\mathbf{x}) = \arg\min_{\mathbf{x}} E_{\text{data}}(f(\mathbf{x}), \mathbf{y}) + \lambda E_{\text{prior}}(\mathbf{x}),$$

where the data term E_{data} measures the similarity between the observation $\mathbf{y}$ and the forward simulated results $f(\mathbf{x})$ of the hidden parameters $\mathbf{x}$; and the prior term E_{prior} encodes humans' knowledge about the solution $\mathbf{x}$.

As the energy function is often non-convex, iterative algorithms are used to refine the estimation. Without loss of generality, the update rule can be written as:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + g_E(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y}), \tag{4.1}$$

where $g_E(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y})$ is an analytical update function derived from the energy function E, and $\mathbf{y}^t = f(\mathbf{x}^t)$. For instance, in continuous-valued inverse problems, $g_E = -A_E(\mathbf{x}^t)\nabla_E(\mathbf{x}^t)$, where $\nabla_E(\mathbf{x})$ is the first-order Jacobian and A_E is a warping matrix that depends on the optimization algorithm and the form of the energy. For instance, A_E is simply a (approximated) Hessian matrix in Newton method and is equivalent to the step size in first order gradient descent.

One major advantage of these approaches is that they explicitly take into account how close $f(\mathbf{x})$ and $\mathbf{y}$ are via the data term E_{data}, and exploit such *feedback* as a guidance for the update. This ensures that the result $f(\mathbf{x}^*)$ generated from the final estimation $\mathbf{x}^*$ is close to the observation $\mathbf{y}$. While impressive results have been achieved, there are several challenges remaining: first, they require both the forward process f as well as the prior E_{prior} to be optimization-friendly (*e.g.* differentiable) so that inference algorithms can be applied. Unfortunately this is not the case for many inverse problems and tailored approximations are required [168, 220, 289, 208, 384]. The performance may thus be affected. Second, they often require many updates to reach a decent solution (*e.g.* first-order methods). If higher order methods are exploited to

Algorithm 1 Deep Feedback Inverse Problem Solver

input observation $\mathbf{y}$, forward model $f(\cdot)$ and init $\mathbf{x}^0$
for $iter = 0, 1, \ldots, T-1$ **do**
    Run forward model: $\mathbf{y}^t = f(\mathbf{x}^t)$
    Compute update: $\mathbf{x}^{t+1} = \mathbf{x}^t + g_{\mathbf{w}}(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y})$
end for
output $\mathbf{x}^T$

speed up the process, the update may become expensive (*e.g.*, second-order methods). Third, carefully designed priors are necessary for identifying the true solution from multiple feasible answers. This is particularly true for ill-posed inverse problems, such as super-resolution and inverse kinematics, in which there exists infinite number of feasible solutions that could generate the observation. Additionally, the energy must be designed in a way that is easy to optimize, which is sometimes non-trivial. Finally, these optimization methods are typically sensitive to the initialization.

4.2.2 Learning based methods

Another line of work [73, 395, 187, 206, 184] has been devoted to directly learning a mapping from the observations $\mathbf{y}$ to the solution $\mathbf{x}$:

$$\mathbf{x}^* = g(\mathbf{y}; \mathbf{w}). \tag{4.2}$$

Here, $g(\cdot; \mathbf{w})$ is a learnable function parameterized by $\mathbf{w}$. These approaches try to capitalize on the feature learning capabilities of deep neural networks to extract statistical priors from data, and approximate the inverse process without the help of any hand-crafted energies. While these methods have achieved state-of-the-art performance in many challenging inverse tasks such as inverse kinematics [273, 438], super-resolution [187, 375], compressive sensing [178], image inpainting [271, 217], illumination estimation [198, 230], reflection separation [429], and image deblurring [251], they ignore the fact that the forward process f is known.

As a consequence, there is no *feedback* mechanism within the model that scores if $f(\mathbf{x}^*)$ is close to $\mathbf{y}$ after the inference, and the model cannot update the estimation

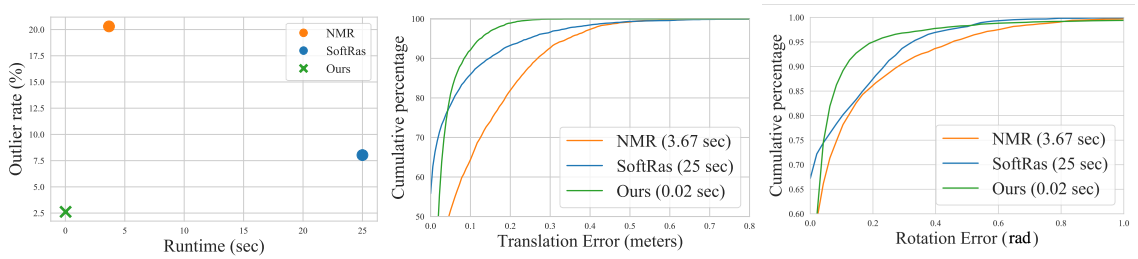


Figure 4-3: **Quantitative analysis on 6 DoF pose estimation:** Our deep optimizer is robust, accurate, and significantly faster.

accordingly. The whole system remains *open loop*.

4.3 Deep Feedback Inverse Problem Solver

In this chapter we aim to develop an extremely efficient yet effective approach to solving structured inverse problems. We build our model based on the observation that traditional optimization approaches and current learning based methods are complementary – one is good at exploiting feedback signals as guidance and inducing human priors , while the other excels at learning data-driven inverse mapping. Towards this goal, we present a simple solution that takes the best of both worlds. We first describe our deep feedback network that iteratively updates the solution based on the feedback signal generated by the forward process. Then we demonstrate how to perform efficient inference as well as learning. Finally, we discuss our design choices and the relationships to related work.

4.3.1 Deep Feedback Network

As we have alluded to above, structured optimization and deep learning have very different yet complementary strengths. Our goal is to bring together the two paradigms, and develop a generic approach to inverse problems.

The key innovation of our approach is to replace the analytical function g_E defined in structured optimization approach at Eq. 4.1 with a neural network. Specifically, we design a neural network $g_{\mathbf{w}}$ that takes the same set of inputs as g_E and outputs

Methods	Optimization Step	Optimization Time	Trans. Error Mean	Trans. Error Median	Rot. Error (°) Mean	Rot. Error (°) Median	Outlier (%)
NMR [168]	105	3.67 s	0.1	0.05	5.78	1.68	20.3
SoftRas [220]	157	25 s	0.05	0.003	4.14	0.5	8.03
Deep Regression	1	0.004 s	0.07	0.06	10.07	7.68	5
Ours	5	0.02 s	0.02	0.009	2.64	1.02	2.6

Table 4.1: **Quantitative comparison on 6 DoF pose estimation.**

the update. The hope is that the model can perceive the difference between the observation $\mathbf{y}$ and the simulated forward results $\mathbf{y}^t$ and then predict a new solution based on the *feedback* signal. In practice, we employ a simple addition rule and fold the step size, parameter priors all into $g_{\mathbf{w}}$:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + g_{\mathbf{w}}(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y}), \quad \text{where } \mathbf{y}^t = f(\mathbf{x}^t). \tag{4.3}$$

The network architectures design depends on the form of observational data $\mathbf{y}$ and solution $\mathbf{x}$. For instance, for inverse graphics tasks, we utilize convolutional neural networks, since the observations are images. This not only allows us to sidestep all requirements imposed on f (*e.g.* differentiability), but also removes the need for explicitly defining energies. Unlike conventional learning based methods, we take both $\mathbf{y}^t$ and $\mathbf{y}$ as input to the update so that we incorporate the feedback signal through comparing the two.

We derive our final deep structured inverse problem solver by applying the aforementioned update functions in an iterative manner. The algorithm is summarized in Alg. 1. At each step, the solver takes as input the current solution $\mathbf{x}^t$, the observation $\mathbf{y}$, and the forward simulated results $\mathbf{y}^t$, and predicts the next best solution as defined in Eq. 4.3. In practice, the stopping criteria could either be based on a predefined iteration number or on checking convergence by measuring the difference between solutions from two consecutive iterations.

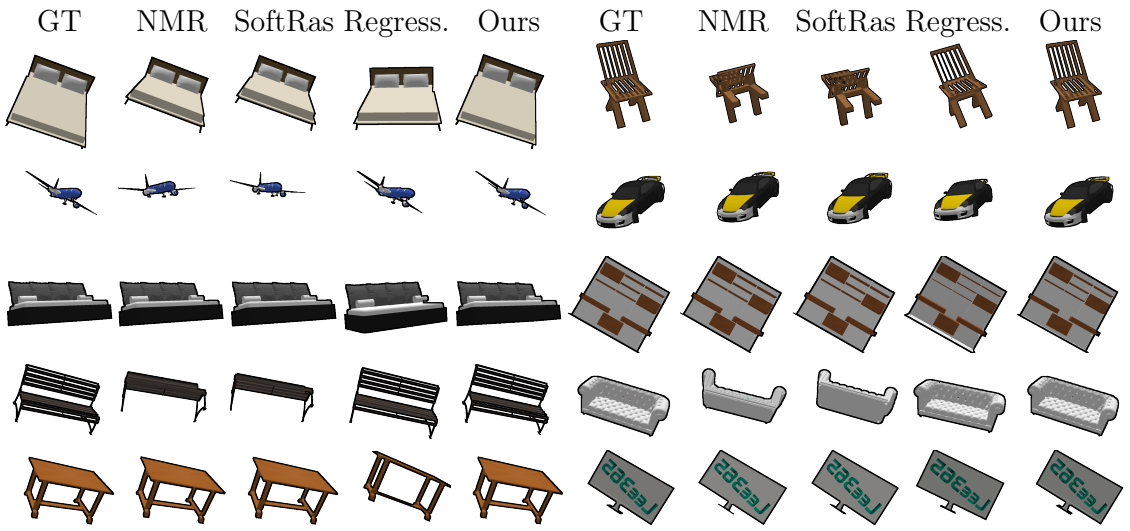


Figure 4-4: **Qualitative comparison on 6 DoF pose estimation:** We infer the poses from only silhouette images. The rendered colored images in the figure are for visualization purpose.

4.3.2 Learning

The full deep structured inverse problem solver can be learned in an end-to-end fashion via back-propagation through time (BPTT). Yet in practice we find that applying loss function over each stage's intermediate solution $\mathbf{x}^t$ yields better results. Deep supervision greatly accelerates the speed of convergence.

However, it is non-trivial to design a learning procedure for each iterative update function $g_{\mathbf{w}}$, as there exist infinite paths towards the ideal solution. Ideally, we would like our solution to descend towards the ideal solution as quickly as possible. Thus, inspired by [394], at each iteration, we learn to aggressively predict the update required to reach the ideal solution. At each stage, the learning procedure finds the best $\mathbf{w}$ through minimizing the following loss function:

$$\arg\min_{\mathbf{w}} \sum_{(\mathbf{y},\mathbf{x}_{\text{gt}})} \sum_{t} \ell(\mathbf{x}_{\text{gt}}, \mathbf{x}^t + g_{\mathbf{w}}(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y})).$$

ℓ is a task-specific loss function; for instance, ℓ is l2-norm for inverse kinematics.

4.3.3 Discussions

Stage-wise network: In our standard approach described before, $g_{\mathbf{w}}$ is shared across all steps. However, the proximity to the ideal solution varies at different step. As a consequence, early iteration often takes inputs that are farther to the ideal solution than what a late iteration update step takes. This brings difficulties to the network as it needs to handle a variety of output scales across different iteration steps. This motivates us to train a separate update function per step $g^t_{\mathbf{w}}(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y})$ that better captures the input data distribution at each iteration. To learn this non-shared weight network, we conduct a stage-wise training procedure. We start to train the $g^0_{\mathbf{w}}$ first. Then we acquire $\mathbf{y}^0$ for all the training data, which allow us to train $g^1_{\mathbf{w}}$. We repeat this procedure until $g^T_{\mathbf{w}}$ is trained. In total T models $\{g^t_{\mathbf{w}}\}$ are trained. Please refer to the supp. material for the comparison between sharing weights and not sharing weights.

Adaptive update: Our current update rule is simply an addition, yet it can be easily extended to more sophisticated settings to handle more complex scenarios. For instance, one can apply the classic momentum technique on top of the predicted gradient to stabilize the optimization trajectory. One can also learn another meta-network to dynamically adjust the output of our update network. While all of these options are feasible, we find that in practice a simple strategy suffices. Inspired by the Levenberg-Marquardt method [33], we exploit a damping factor λ to control the effectiveness of the update network, *i.e.*, $\mathbf{x}^{t+1} = \mathbf{x}^t + \lambda \cdot g_{\mathbf{w}}(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y})$. Specifically, λ is initialized to 1 at the beginning of each update. If the new estimation results in a lower data energy than that of the original one, we update the estimation. Otherwise we reduce λ by half and re-compute. We only need to compute the update gradient once. The forward process is executed on the GPU and hence the computational overhead is negligible. Through this simple rule, we can guarantee that $E_{\text{data}}(\mathbf{x}^t, \mathbf{y})$ decreases after every iteration. Empirically $\mathbf{x}^t$ becomes closer to the ground truth $\mathbf{x}$ as well, since the ambiguity arising from the data term disappears when the estimation is already sufficiently close.

Relationship to existing work: Our model is closely related to the family of iterative networks [109, 349, 201, 39, 105, 285, 347, 382, 381, 47, 205, 207], in particular the stacked inference machines [285, 347, 382, 381]. Unlike previous methods that require the model to implicitly learn the relationship between the input and the preceding estimation [262, 47, 381, 415, 236], we leverage the forward process to explicitly establish the connection among them and close the loop. This is of crucial importance for inverse problems since the two spaces are very distinct (*e.g.* illumination parameters vs RGB image). The idea of learning to update is inspired by supervised descent methods [394]. However, unlike their approach we learn the mapping and the feature simultaneously. Furthermore, we focus on inverse problems and design a closed-loop scheme to incorporate feedback signals, while they simply perform iterative update in an open loop setting. Developed independently, Flynn *et al.* [100] propose a similar approach for view synthesis. Their model, however, relies on the analytical gradient components. They thus requires the system to be differentiable. In contrast, our approach directly predicts the update from the observation and the feedback signal. We do not require explicit gradient computation and do not have such a limitation. Similar to our work, LiDO [302] also leverages deep networks to optimize the latent parameters. Yet unlike their model which directly regresses the GT, we predict the residual instead. This is very critical as the magnitude of the update is strongly correlated with the input difference. The larger (smaller) the input difference is, the more (less) update is required. Predicting the residual can thus significantly ease the learning process. Furthermore, while their prediction networks share weights across all iterations, our feedback networks differ at each step. We are able to model the variety of output scales more effectively. Our work also shares similar insights as [262, 47, 194] with a few key differences: (1) rather than relying on the network to *implicitly* establish the relationships between the feedback signal and the observation, we *explicitly* consider the difference and predict an update based upon it. (2) Our model is motivated by classic optimization approaches. We borrow ideas from traditional literature to improve the performance (*i.e.* adaptive update), whereas [194] simply unrolls the network, [47] employs a bounded, fixed update, and

Module	Forward Rendering	Inverse Update	Total
NMR [168]	28 ms	7 ms	35 ms
SoftRas [220]	76 ms	84 ms	160 ms
Ours	2.6 ms	0.9 ms	3.5 ms

Figure 4-5: **Runtime breakdown of a single optimization step for 6 DoF pose estimation.**

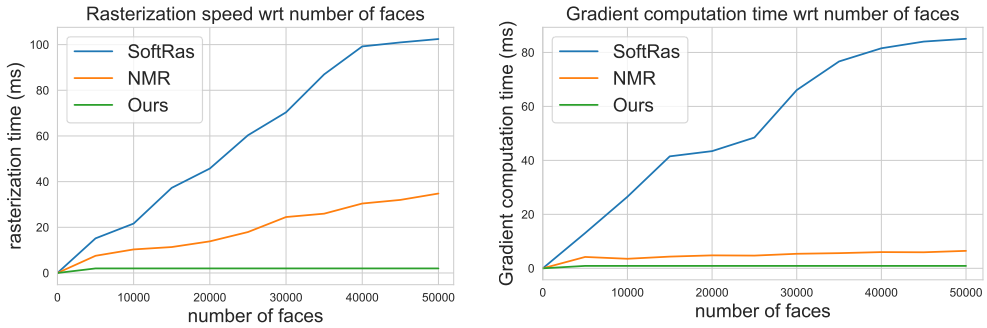


Figure 4-6: **Runtime vs number of faces.** (Left) Forward rasterization time. (Right) Backward gradient computation (inverse update) time.

[262] encourages the update network to improve the estimation, no matter what scale it is. (3) We present a generic framework that is applicable to a wide range of inverse problems, while [262, 47, 194] are specialized to respective specific task. We refer the readers to the supp. material for more detailed discussions on reinforcement learning and prior art [47, 194, 302, 262].

Applicability: Unlike previous work, our approach neither has restrictions on the forward process f, nor need to construct domain-specific objectives at test time. During inference, at each iteration, we simply adopt a feed-forward operation g on top of current estimate and predict the update. Our method is applicable to a wide range of tasks so long as the forward process function f is available. In the following sections, we showcase our approach on two different inverse graphics tasks (object pose estimation and illumination estimation from a single image) as well as one robotics task (inverse kinematics).

Training on $0° - 40°$	Trans. Error		Rot. Error (°)	
Evaluation Rot. Range	Mean	Median	Mean	Median
$40° - 45°$	0.05	0.03	11.33	4.97
$45° - 50°$	0.05	0.04	15.62	5.60
$50° - 55°$	0.06	0.04	18.58	6.86
$55° - 60°$	0.07	0.05	24.14	9.58

Table 4.2: **Test on unseen rotations.**

4.4 Application I: 6-DoF Object Pose Estimation

Problem formulation: Assume that the 3D model of the object is given [132, 45] and the camera intrinsic parameters are known. For a given object pose wrt the camera, denoted as $\mathbf{x} \in \mathrm{SE}(3)$, we can generate the corresponding image observation $\mathbf{y}$ through a forward rendering function $f : \mathbf{x} \to \mathbf{y}$, powered by a graphics engine. The goal of 6 DoF pose estimation is to *invert* the process and recover the latent pose $\mathbf{x}$ from the observation image $\mathbf{y}$. This problem is particularly important for problems such as robot grasping [194] and self-driving [232]. Unlike previous approaches that leverage RGB information or depth geometry to guide the pose estimation, we focus on a more challenging setting where the observation is *a single silhouette image* $\mathbf{y} \in \{0, 1\}^{H \times W}$. The object pose $\mathbf{x} = (\mathbf{x}_{\text{quat}}; \mathbf{x}_{\text{trans}})$ is represented by a unit quaternion for rotation $\mathbf{x}_{\text{quat}}$ and a 3D translation vector $\mathbf{x}_{\text{trans}}$.

Data: We use the 3D CAD models from ShapeNet [51] within 10 categories: cars, planes, chairs, bench, table, sofa, cabinet, bed, monitor, and couch. The dataset is split into training (70%), validation (10%) and testing (20%). For each object, we randomly sample an axis from the unit sphere and rotate the object around the axis by $\theta \sim [-40, 40]$ degrees. We further translate the object along each axis by a random offset within $[-0.2, 0.2]$ meters. Given the randomly generated ground truth object poses, we render 128×128 silhouette images with non-differentiable PyRender [1] as input observations. We refer the readers to the supp. material for the performance of our model on other image sizes.

Metrics: We measure the translation error with euclidean distance and the rotation error with angular difference. Inspired by [102], we also compute the *outlier ratio* as an indicator of the general quality of the output. Specifically, we define the prediction to be an outlier if the translation error is higher than 0.2 or the rotation error is larger than 30°.

Network architecture: Our deep feedback network $g_{\mathbf{w}}$ is akin to the classic LeNet [186]. It takes as input the rendered image $\mathbf{y}^t = f(\mathbf{x}^t)$, the observed image $\mathbf{y}$, as well as the difference image $\hat{\mathbf{y}} - \mathbf{y}^t$, and directly outputs the update $\Delta\mathbf{x}$. We apply an additional normalization operator over the rotation component to correct it to a valid unit quaternion. We unroll our deep feedback network for five steps. MSE is employed as the loss function for both rotation and translation since it produces the most stable results.

Baselines: For optimization methods, the energy function consists of a data term $E_{\text{data}}(f(\mathbf{x}), \mathbf{y})$ that favors agreement and a prior term $E_{\text{prior}}(\mathbf{x})$ that encourages the quaternion to remain on the manifold. To make the forward rendering procedure f differentiable, we utilize the state-of-the-art differentiable renderers for comparison, *i.e.* neural mesh renderer (NMR[168]) and soft rasterization (SoftRas [220]). We utilize the following stopping criteria for the optimizer: (i) 500 iterations, or (ii) the silhouette difference between the observation and the one generated by the renderer stops improving for 20 iterations. For the deep regression method, we use the same architecture as our deep feedback network except that no feedback is provided.

Results: As shown in Tab. 4.1, our method achieves a significantly lower outlier ratio compared to other approaches. This indicates that our model is more robust and less susceptible to becoming stuck in local optimum. It also has comparable performance to differentiable renderers in terms of mean translation and angular error, while being two to three orders of magnitude faster. On the other hand, our method has much better performance than the non-feedback deep regression method. For the category-wise performance, please refer to the supp. material. Fig. 4-4 showcases some

Methods	Optimization # of steps	Optimization Time	Directional light Mean	Directional light Median	Directional light Outliers	Point light Mean	Point light Median	Point light Outliers
NMR* [168]	166.7	58.3 s	0.099	0.037	19.2%	-	-	-
Deep regression [151]	1	0.043 s	0.067	0.022	24%	0.111	0.084	11%
Ours	7	0.183 s	0.052	0.008	8%	0.084	0.064	9%

Table 4.3: **Illumination estimation on ShapeNet.**

qualitative results. Our method is robust to extreme poses, whereas optimization based method is easy to get stuck in a local optimum.

Deep feedback network as initialization: Due to the highly non-convex structure of the energy model, a good initialization is required for optimization methods to achieve good performance. One natural solution is to exploit our model as an initialization and employ classic solvers for the final optimization. By combining our approach with SoftRas, we can further reduce the error by more than 50%. We refer the readers to supp. material for detailed analysis.

Runtime analysis: We show the runtime break down for a single update step in Tab. 4-5 and the runtime w.r.t the number of faces in Fig. 4-6. As we neither need to construct the computation graph nor storing any activation value for gradient computation during the forward rasterization process, our rendering is significantly faster. For gradient computation, SoftRas is far slower as it needs to propagate the gradient to multiple faces. In contrast, our update model is simply an efficient feed-forward neural net that takes as input the (difference) silhouette images. Its speed is invariant to the number of faces.

4.5 Application II: Illumination Estimation

Problem Formulation: We next evaluate our method on the task of illumination estimation. The goal is to recover the lighting parameter $\mathbf{x} \in \mathbb{R}^3$ from the observation RGB image $\mathbf{y} \in \mathbb{R}^{H \times W \times 3}$. It has critical applications in image relighting and photo-realistic rendering [167]. As in the 6-DoF pose estimation task, we assume the 3D model is given.

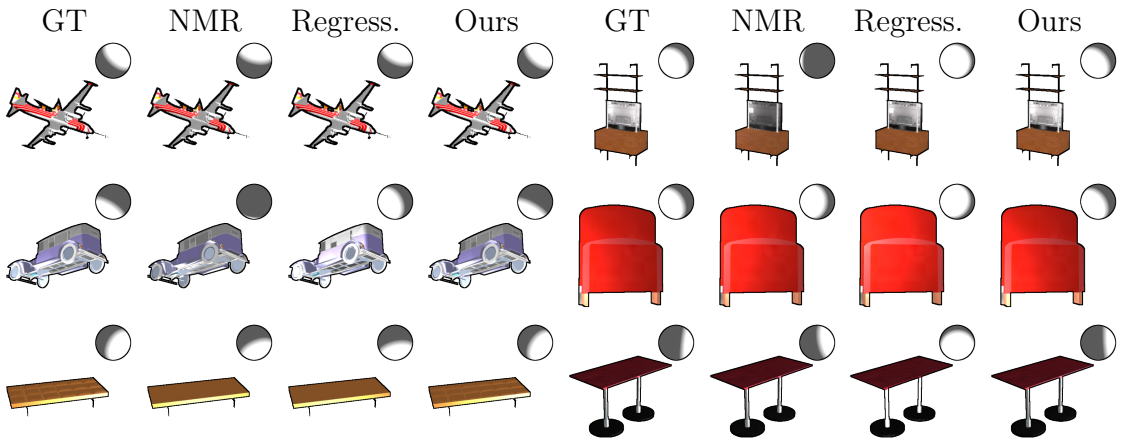


Figure 4-7: **Qualitative comparison on illumination estimation.**

Data: We use the same dataset as the 6-DoF pose estimation experiment for the illumination estimation experiment. Specifically, we consider two types of light source: directional light and point light. The two light sources are complementary and can result in very different rendering effect. During training, we randomly sample the light position from the half unit sphere on the camera side [151, 230]. If the light is directional, we point the light towards the origin. All the objects are set to have Lambertian surfaces. We ignore the scenario where the light source lies on the other side of the object, as it has no effect on the rendered image. For evaluation, we follow the same criteria. We perform rendering in pyrender and the image size is set to 256×256. Empirically we found this size provides the best balance between performance and the computational speed.

Metrics: Following [151], we use the standard mean-squared error (MSE) between the ground truth light and estimated light pose to measure the difference. We also compute the outlier rate as described in Sec. 4.4.

Network architecture: We employ an encoder-decoder architecture with skip connections as our deep feedback network. Since the 3D geometry of the object plays an important role during rendering, we adopt depth prediction as an auxiliary task. This allows the model to implicitly capture such notion and reason about its

relationship with illumination. During training, our deep feedback network estimates both the depth of the object as well as the illumination parameters. We use MSE as the objective for both tasks. During inference, we simply discard the depth decoder and output only the illumination part. We unroll our network for 7 steps according to the validation performance.

Baselines: We exploit NMR [168] to minimize the energy $E_{\text{data}} + E_{\text{prior}}$.The data term is the ℓ_2 distance between the observation image and the rendered image, while the prior term constrains the light source to lie on the sphere. We adopt the same stopping criteria as in Sec. 4.4. The size of the rendered image is set to 256×256 based on the performance on the validation set. For deep regression method, we exploit the state-of-the-art model from Janner *et al.* [151].

Results: As shown in Tab. 4.3, our deep feedback network outperforms the baselines on both setup. The improvement is significant especially in the directional light case. We conjecture this is because the intensity of directional light does not decay w.r.t. the travel distance, and the signals from the image are weaker. Learning based approaches thus have to rely on feedback signals to refine the light direction. The performance of the optimization method is limited by the hand-crafted energy as well as the capability of renderer. NMR is sub-optimal as it approximates the gradient with a manually designed function and does not handle self-occlusion. In contrast, our method allows us to exploit complex rendering machines as the forward model as we do not require it to be differentiable. We note that we only report the optimization results on directional light since NMR does not support point light source. Fig. 4-7 depicts the qualitative comparison against the baselines. It is clear that our deep feedback mechanism is able to recover accurate lighting information based on subtle difference between the forward results and the observations.

Methods	Optimization Step	Optimization Time	Position Error (cm) Mean	Position Error (cm) Median	Rotation Error (°) Mean	Rotation Error (°) Median
L-BFGS [110]	73	27.9 s	0.38	0.01	7.19	4.68
Adam [173]	196	38.8 s	0.04	0.04	7.96	7.92
Deep6D [438]	1	0.012 s	1.9	1.6	-	-
Ours	4	0.12 s	0.64	0.36	0.88	0.03

Table 4.4: **Quantitative results on CMU MoCap.**

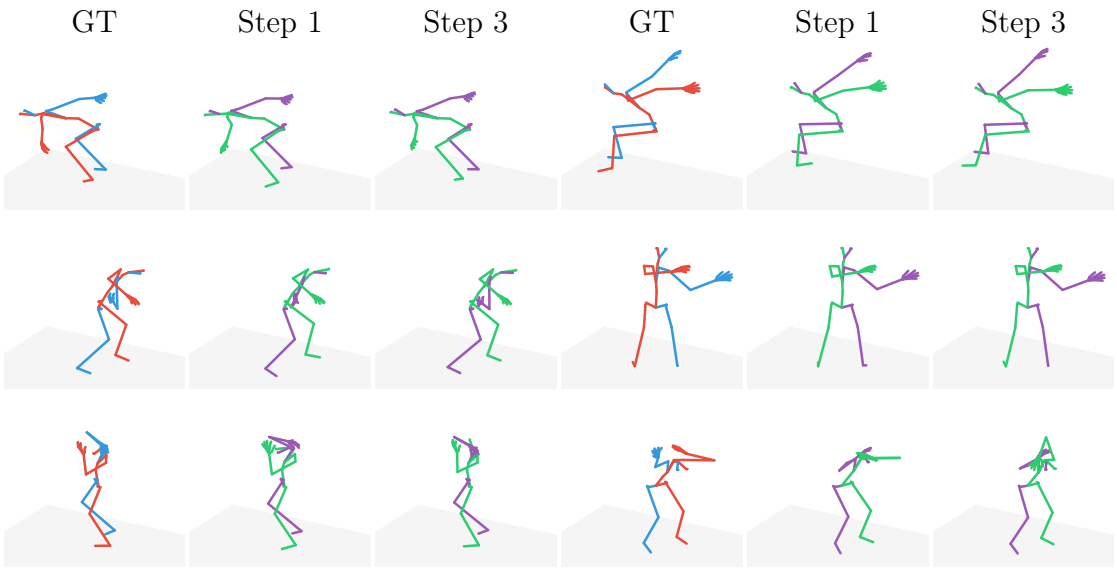


Figure 4-8: **Qualitative results on CMU MoCap:** Our approach is able to accurately predict the joint rotations within a few steps. It can also correct wrong estimations through the feedback from the forward model (see the feet/toes in the right column). Bottom right shows an example where our model fails.

4.6 Application III: Inverse Kinematics

Problem formulation Finally we exploit how our proposed method to tackle the inverse kinematics problem. Given the 3D location of the joints of a reference pose $\mathbf{y}^{\text{ref}}_{1:N}$ and the desired joint rotations $\mathbf{x}_{1:N} \in \text{SO}(3)$, the forward kinematics function f rotates the joints and computes their 3D positions by recursively applying the follow update rule from parents to children: $\mathbf{y}_n = \mathbf{y}_{\text{parent}(n)} + \mathbf{x}_n(\mathbf{y}^{\text{ref}}_n - \mathbf{y}^{\text{ref}}_{\text{parent}(n)})$. The goal of inverse kinematics is to recover the SO(3) rotations $\mathbf{x}_{1:N}$ that ensure the specific joints are placed at the desired 3D locations $\mathbf{y}_{1:N}$. Inverse kinematics has a wide range of applications, such as robot arm manipulation, legged robot motion planning and computer re-animation. The problem is inherently ill-posed as different rotations

can result in the same observation through the forward kinematics function f, *i.e.*, $\mathbf{y} = f(\mathbf{x}_{1:N}) = f(\mathbf{x}'_{1:N})$. However, not all angles are feasible or natural due to the dynamic constraints. Therefore, in order to accurately recover the rotations, one has to either come up with a powerful prior or learn it from data. In this chapter, we focus on inverse kinematics over human body skeletons.

Data: We validate our model on the CMU Motion Capture Dataset (CMU MoCap) as it contains complex human motions and a diverse range of joint rotations. Following Yi *et al*. [438], we select 865 motion clips from 37 motion categories and hold out 37 clips for testing. Each skeleton in the dataset has 57 joints. We fix the position of the hip to remove the effect of global motion.

Metrics: We evaluate the performance of our model with joint position error [438] and joint angular error [240, 273]. The two metrics are complementary since a small rotation error may result in a large position error due to the recursive nature of the forward kinematics model, and small position error cannot guarantee correct joint rotation due to ambiguities.

Network architecture: Our deep feedback network is a multilayer perception akin to [438]. Following [357, 273], the network takes as input the estimated joint position, reference joint position, as well as the difference between the two, and outputs a rotation for each joint. We unroll our model three steps. We train the network with L2 loss on both position error and rotation error.

Baselines: We compare our model against two optimization-based approaches and one deep regression method. For optimization methods, we employ joint position error as our data term, *i.e.* $E_{\text{data}}(f(\mathbf{x}), \mathbf{y}) = \|f(\mathbf{x}) - \mathbf{y}\|_2^2$, and derive a prior energy term from data to alleviate the ambiguities of joint rotations. In particular, we fit a gaussian distribution over the Euler angles of each joint from training data and employ it as a regularization term during inference. We set the weight of the prior term to 0.001 and optimize both energies jointly. We exploit two different types of

optimizers: a first-order method (*i.e.*, Adam [173]) and a quasi-Newton method (*i.e.*, L-BFGS [110]). For deep regression method, we compare with the current state of the art (Deep6D [438]).

Results: As shown in Tab. 4.4, our deep feedback network outperforms the baselines on the rotation metric and achieve comparable performance on the position error. By unrolling more steps and gathering feedback signals from the forward model, we are able to reduce incorrect estimation and improve the performance (see the Fig. 4-8). We refer the readers to the supp. material for detailed analysis. On average, a single step of L-BFGS, Adam, and our approach takes 383 ms, 198 ms, 30 ms respectively. L-BFGS takes longer to compute as it needs to conduct gradient evaluation multiple times to approximate the Hessian. Adam is faster in terms of computation, yet it takes far more steps to converge. Our approach, in comparison, is significantly faster and better.

4.7 Conclusions

In this chapter, we propose a deep feedback inverse problem solver. Our method combines the strength of both learning-based approaches and optimization-based methods. Specifically, it learns to conduct an iterative update over the current solution based on the feedback signals provided from the forward process of the problem. Unlike prior work, it does not have any restrictions on the forward process. Further, it learns to conduct an update without explicitly define an objective function. Our results showcase that the proposed method is extremely effective, efficient, and widely applicable.

Chapter 5

Exploiting High-level Priors for 3D Reconstruction

Virtual Correspondence:
Humans as a Cue for Extreme-View Geometry
Wei-Chiu Ma, Anqi Joyce Yang, Shenlong Wang, Raquel Urtasun, Antonio Torralba;
CVPR 2022.

In previous chapters, we discussed how to exploit various low-level structures, such as geometry and physics, to develop robust computational models for 3D understanding. While many 3D modeling tasks are indeed considered low-level (*e.g.* 3D reconstruction), in this chapter we take a different approach. We study the problem through the lens of high-level vision. We investigate how a high-level understanding of an object can benefit low-level 3D reconstruction and significantly expand the application domain of existing 3D systems.

5.1 Introduction

Epipolar geometry and correspondence estimation are two keystones of mainstream 3D reconstruction systems. When given a set of RGB images as input, a classic 3D

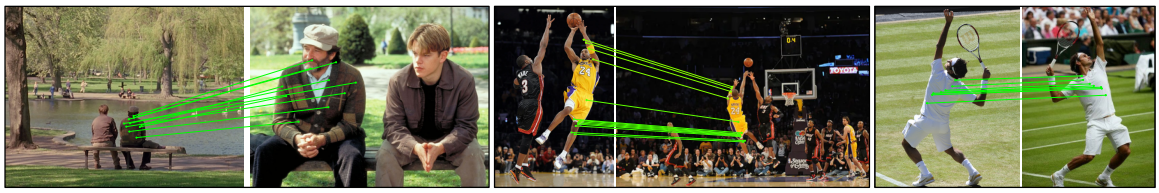


Figure 5-1: **Can you tell the relationships between these matched pixels?** The head pixel and the face pixel in the leftmost images have completely different semantics and appearances, yet we can still associate them for 3D reasoning. Why, and how? In this chapter, we present a novel concept to establish geometric relationships between pixels even if they are not semantically or visually similar. See Fig. 5-11 for cars.

pipeline [224, 121] first identifies *co-visible* 3D points across images via pixel-wise visual features and then recovers the spatial relationships among cameras. Such a "golden standard" framework has experienced huge success in practice and has given birth to numerous applications in robotics, AR, VR, etc. The reliance on correspondences, however, makes one ponder: what if the input images have little or no overlap? Does this still work when there are barely any co-visible 3D points in the scene (see Fig. 5-1)?

At first thought the answer is no. Predominant correspondence estimators focus on finding pixel pairs that describe the same, co-visible 3D points in the scene by matching their visual features. If the viewpoint differences across images are extreme, the pixels will be inherently different and cannot be matched, causing current 3D systems to fail catastrophically. In contrast, humans can identify where two photographs were taken with respect to the same scene, despite large viewpoint variations. Such a remarkable capability comes from our prior knowledge of the underlying geometry, which helps us match pixels between images even if their exact correspondences are occluded or invisible in the other image. For instance, we know what the front and back of a human body should look like. Therefore, if we see a human face in one image and the back of a head in the other, we can easily associate them and infer that the two cameras are roughly 180 degrees apart. The aim of this chapter is to equip 3D systems with similar abilities.

Towards this goal, we first ask the following question: *do we have to rely on*

pixels describing the same 3D points to recover camera poses[*]? While such (implicit) premises seem to lay the foundation for existing 3D reconstruction algorithms, as we will show in Sec. 5.3, the answer is negative. Our key observation is that epipolar geometry holds for *arbitrary pixels whose camera rays intersect in 3D*. Therefore, as long as we can identify those pixels, we can leverage them to recover relative camera poses, *regardless of whether the pixels are semantically or visually similar or not*. This interpretation is particularly exciting, as it allows us to go beyond the image space and establish geometric relationships among pixels even from extreme viewpoints.

Unfortunately, determining whether two camera rays intersect in 3D often requires camera poses to be known a priori, making the whole process a chicken-and-egg problem. Our key idea is to exploit prior knowledge of the foreground objects within the scene to break the loop. Specifically, we make use of *humans*, arguably one of the most common, salient "objects" in images. Consider the images in Fig. 5-1. If the system has prior knowledge about human shape and pose, it will know that a ray shooting through the human back in the leftmost image will intersect with the chest region on its way out. Furthermore, the intersecting chest pixel can be observed in the other image. Thus, we can find a pair of pixels that correspond to two intersecting camera rays with ease. Note that different from classic correspondences, these two pixels do not depict the same 3D point and thus cannot be found via visual similarities. Since we establish the geometric connection virtually by hallucinating a 3D shape, we call them *virtual correspondences* (VCs).

With this inspiration in mind, we first define virtual correspondences and present a methodology to derive them from images containing humans. We then showcase how VCs can be seamlessly integrated with the classic bundle adjustment algorithm, resulting in a generalized structure from motion (SfM) framework that could be applied to both traditional setup and extreme-view scenarios. We evaluate the effectiveness of our approach on the CMU Panoptic dataset [155, 156], the Mannequin Challenge dataset [199], and multiple challenging in-the-wild images. Our method significantly outperforms prior art in challenging extreme-view scenarios and is comparable in

[*]We will ignore other primites such as lines or planes for now.

the conventional, densely overlapping setup. Importantly, our estimated poses from extreme viewpoints unleash the potential of multiple downstream applications such as scene reconstruction from multi-view stereo and novel view synthesis in challenging scenarios.

In summary, we make the following contributions:

1. We present virtual correspondences, a novel concept for 3D reconstruction algorithms, and establish its geometric connection to existing correspondences.

2. We develop a method to estimate VCs from images with humans and showcase how to integrate them into existing 3D frameworks. The new framework can be applied to a wide range of scenarios while also reduces to the classical SfM when no VCs are found.

3. We exploit the estimated camera poses for multiple downstream applications and empirically show that our method can extend these tasks to extreme-view scenarios which were previously infeasible.

5.2 Related Work

Correspondences: Correspondence estimation aims to identify pixels that are projections of the same 3D point across multiple images [121, 233]. The task has been the cornerstone of various computer vision problems for decades, since the pixel-level association allows one to recover the structure and motion of the world effectively [224, 138, 26, 313]. Prevalent approaches focus on hand-crafted [226, 42, 189, 345, 25, 305] or learned [370, 368, 264, 72, 374, 83, 365] robust *visual* features that can distinguish one pixel from the others in diverse scenarios. While impressive performance has been achieved [308, 334], these methods fall short when there is little overlap among input images, as there are hardly any co-visible 3D points. Semantic correspondence estimation [119, 435, 437, 64, 172, 120, 144], on the other hand, focuses on detecting pixels with specific semantics (*e.g.*, human facial keypoints). With the help of domain knowledge, they are usually more robust to variations in viewpoint, appearance, and

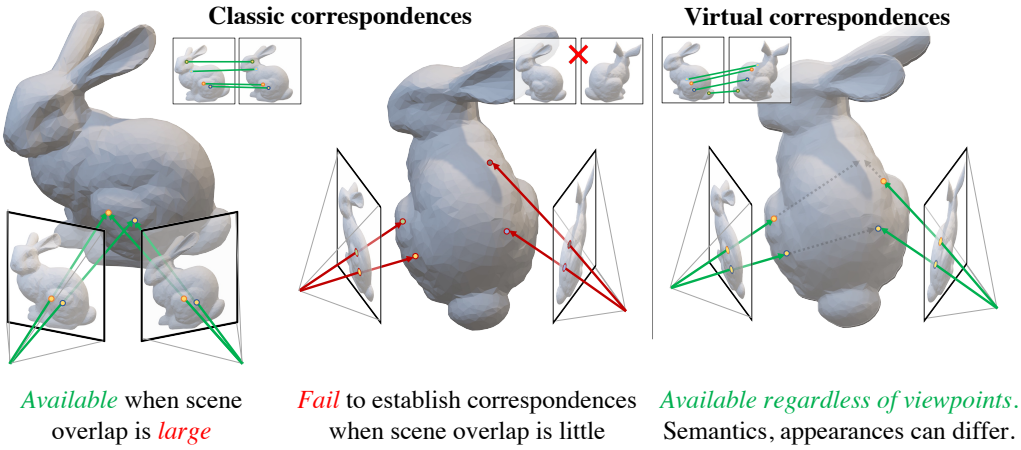


Figure 5-2: **Classic correspondences vs. virtual correspondences.**

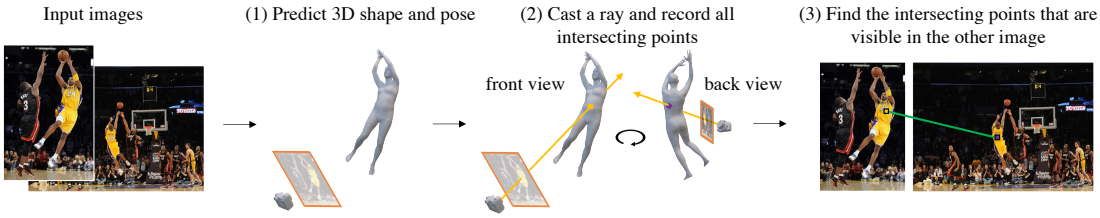


Figure 5-3: **Pipeline of estimating virtual correspondences.** We first predict the 3D shape and pose of the basketball player from the left image. Then we cast a ray and record all the points it hits, *i.e.* the belly button and his back. While the two images barely overlap, the right image does observe the back of player. We can thus tell that the rays of the two pixels intersect at 3D and are virtual correspondences. We conduct the same process for the right image too.

sometimes even occlusions [140, 44]. Unfortunately, they still require a set of semantic keypoints to be co-visible across multi-view images to enable 3D reconstruction. In contrast, our novel virtual correspondences do not have these constraints. VCs can be the projection of different 3D points and can have completely different appearances and semantics (*e.g.*, chest pixel *v.s.* back pixel). This allows us to establish geometric relationships among pixels even when the input images have no co-visible 3D points.

Extreme pose estimation: There has been a surge of interest in estimating relative 3D poses among a set of little- or non-overlapping RGB(D) images [404, 279, 41, 153, 318]. Different from the classical small- or wide-baseline setup, the large viewpoint variations in this task result in very few co-visible regions, rendering traditional

matching-based approaches unsuitable. To address this challenge, researchers have proposed to either directly predict the transformation with deep neural nets [41, 54], or adopt the hallucinate-then-match paradigm [404, 406, 279, 17, 103]. Our work lies under the broad umbrella of the hallucination paradigm, as we derive virtual correspondences from hallucinated human shape priors and combine them with epipolar geometry. We adopt a pixel-level correspondence representation, which seamlessly integrates with prevailing 3D reconstruction algorithms and can be naturally extended to the multi-view setup. In contrast, previous methods only consider two frames at a time [404, 406, 279, 153], as the customized matching and optimization step prohibits them from scaling up easily.

Structure from motion (S*f*M): Given a set of images, the goal of SfM algorithms [26, 336, 65, 98, 346, 348, 275, 4] is to recover both the camera poses and the (sparse) 3D geometry of the scene. Prevailing SfM systems [314, 326, 327, 388] have enjoyed great success when the images are captured densely with large overlapping regions, yet they suffer drastically when the input views are sparse and have little overlap. To alleviate this issue, researchers have sought to exploit motion patterns [10, 11, 335] or semantic keypoints of the objects [74, 396] to aid the reconstruction. However, they require sequences of frames as input (with static cameras) or the same set of keypoints to be visible across all views, which largely limits their applicability. Our virtual correspondences, in comparison, are much more flexible: while our VCs are also derived from objects, specifically humans, the corresponding pixels can have completely different semantics and appearances. This allows us to establish matches even if the input images have no co-visible 3D points. Our approach also shares similar insights with non-rigid SfM algorithms, which leverage shape dictionaries (*i.e.*, priors) to constrain the solution space [35, 70, 6, 69, 177, 152, 360]. However, unlike these approaches, we do not require 2D correspondences to be given a priori. We instead exploit shape priors to establish VCs across views that conventionally do not have correspondences. As we will show in the experimental section, VCs open the door to a range of possibilities and broaden the applicable domain of SfM.

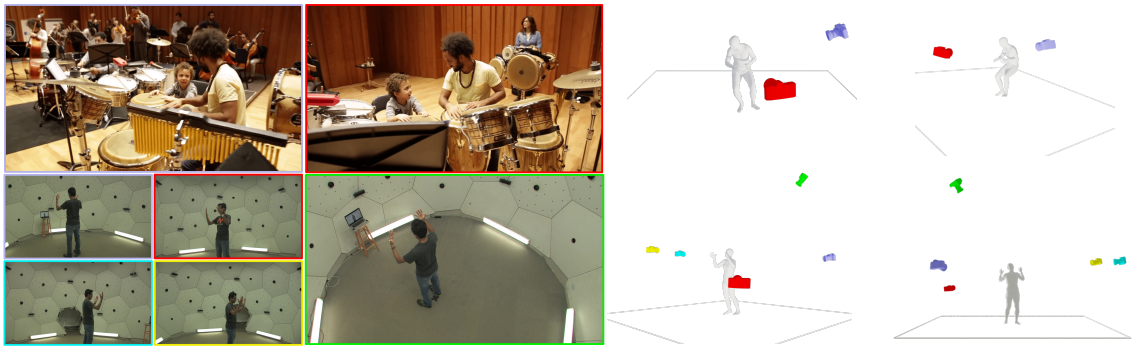


Figure 5-4: **Qualitative results.** (Left) Input images. (Right) Recovered camera poses. Human meshes are for illustration purposes.

3D human estimation: Our work is also related to 3D human reconstruction approaches [118, 400, 401]. With the rise of deep learning, these methods have made tremendous progress, either from a single image [162, 176, 158] or multi-view images [272, 74, 75, 96]. While these approaches mostly focus on the quality of the reconstructed shape, we attempt to recover accurate camera poses with human shape priors. More recently, researchers have exploited human keypoints to refine camera poses [278, 74], but by virtue of VCs, our method is more flexible and does not require the same keypoints to be co-visible across views. As we will show in Sec. 5.3.3, our bundle adjustment formulation is a superset of theirs. Our work also shares similar insights with human silhouette matching [324, 323], since we both do not rely on appearance matching to establish correspondences, allowing us to generalize to extreme-view setting. However, there exist several differences: First, while they require video sequences to constrain the solution space, a single image pair suffice for us. Second, they capitalize on sufficient motion of the object over the space for matching, whereas we exploit deep shape priors to estimate the correspondences. Third, their frontier points are still co-visible across cameras, yet our VCs may correspond to completely different 3D points.

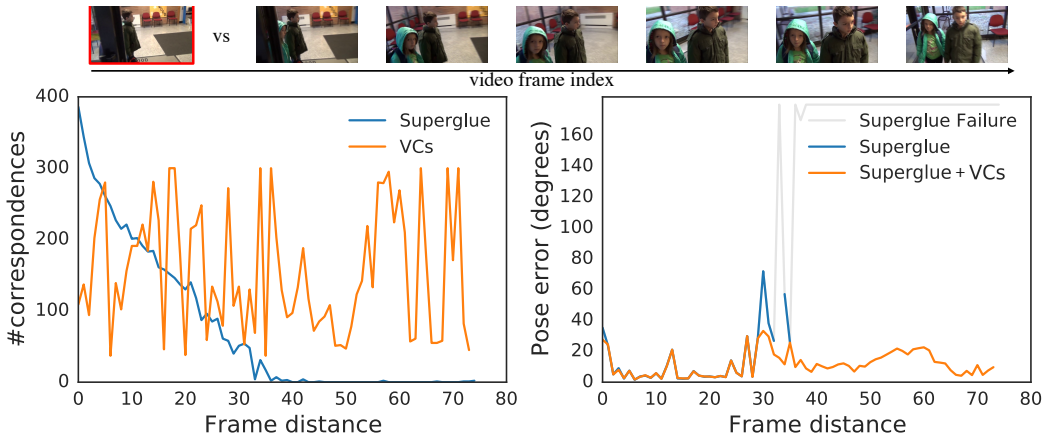


Figure 5-5: **Effects of camera distance.** We show the number of correspondences (left) and pose error (right) with increasing camera baseline.

5.3 Approach

Our aim is to equip existing 3D systems with the ability to reason and associate images geometrically even if they have little or no overlap. We seek to devise a method that can be seamlessly integrated with existing 3D reconstruction frameworks such that the new model can be applied to both the conventional setup and the extreme setting. Towards this goal, we introduce a novel concept dubbed as *virtual correspondence (VC)*. VCs refer to a pair of pixels whose camera rays intersect in 3D. However, unlike classic correspondences, they do not need to describe the same 3D points, and can have completely different semantics and appearances. This makes VCs much more flexible and allows VCs to be established even when there is little overlap among images. Importantly, VCs conform to epipolar geometry and can be combined with prevailing 3D systems naturally. We unfold this section by formally defining VCs and discussing their relationships with existing correspondences. Then we present a method to estimate VCs through the lens of human shape priors. Finally we incorporate VCs into current SfM formulations, resulting in a framework that is much more general. For simplicity, we assume there are only two cameras, but the concepts and the method can be trivially extended to the multi-camera setup (as shown in Sec. 5.4).

5.3.1 Virtual Correspondences (VCs)

We first define virtual correspondences. Let $\mathcal{I}_1$, $\mathcal{I}_2 \in \mathbb{R}^{H\times W\times 3}$ be the images of the same scene captured at different viewpoints and $\boldsymbol{p}_1$, $\boldsymbol{p}_2 \in \mathbb{R}^2$ be the points in their respective image coordinates. Let $\mathbf{K}_1, \mathbf{K}_2 \in \mathbb{R}^{3\times 3}$ be the camera intrinsics and $[\mathbf{R}_1, \mathbf{t}_1], [\mathbf{R}_2, \mathbf{t}_2] \in \mathbb{R}^{3\times 4}$ be their extrinsic matrices. The ray marching from the camera center $\mathbf{o} \in \mathbb{R}^3$ through $\boldsymbol{p}$ can be written as $\boldsymbol{r}_{\boldsymbol{p}}(d) = \mathbf{R}^T(d\mathbf{K}^{-1}\bar{\boldsymbol{p}} - \mathbf{t})$, where $d > 0$ indicates the depth along the ray and $\bar{\cdot}$ refers to the homogeneous coordinate.

We say a point $\boldsymbol{p}_1$ in the first image and a point $\boldsymbol{p}_2$ in the second image, $(\boldsymbol{p}_1, \boldsymbol{p}_2)$, are *virtual correspondences* if there exists a pair of d's such that:

$$\boldsymbol{r}_{\boldsymbol{p}_1}(d_1) = \boldsymbol{r}_{\boldsymbol{p}_2}(d_2). \tag{5.1}$$

Since there is no constraint on where the intersection should happen, the rays can intersect at (i) co-visible 3D points, (ii) 3D points that are only visible in one image (and occluded in other other), or even (iii) invisible points (*e.g.*, free space, occupancy space, or points from occluded scene/objects).

The first scenario is exactly the definition of classic correspondences [72, 308]. The third scenario covers many cases in semantic correspondence where the target 3D points is invisible. For instance, researchers have exploited 2D human keypoints to reconstruct 3D joints [74, 54]. 3D joints, strictly speaking, lie within the human body and are not visible in images. VCs can therefore be seen as a generalization of multiple types of existing correspondences.

In the second and third scenario, VCs correspond to different 3D points in the scene. VCs can thus have different appearances and semantics, and even describe completely different parts of the scene. We show an example in Fig. 5-2 (right) where the pixels in the left image observe the leg while their VCs in the right see the back of the bunny. We refer the readers to supp. material for more illustrations.

Another key property of VCs is that they conform to epipolar constraints — the two intersecting rays form an epipolar plane on which the VCs and camera origins lie. This allows us to exploit classic geometric algorithms to establish connections

among non-overlapping images, greatly expanding the applicable domains of existing 3D algorithms. For instance, we cannot employ the five-point algorithm [224] for non-overlapping images in the past, since no correspondences exist. VCs, however, are more flexible and are not restricted to describing the same co-visible scene points. We can thus estimate VCs among the images and then solve for the essential matrix. We refer the readers to supp. material for more discussion on VCs and epipolar geometry.

While VCs are powerful, estimating them purely from 2D images is far from trivial. Without knowing the relative camera poses, one cannot exploit Eq. 5.1 to verify if two camera rays intersect. Furthermore, VCs may have completely different appearances and semantics, prohibiting us from employing similar approaches as classic correspondence estimators. Fortunately, there are many objects in the scene whose shapes we are familiar with. With such prior knowledge, we can hallucinate the shape of an object and estimate which part of the object a ray would intersect with on the other side. All one needs to do is then to find the rays (pixels) in other images that hit (see) the same intersecting point.

5.3.2 Exploiting Humans for VC Estimation

Based on the intuition above, we propose an approach to exploit shape priors for virtual correspondence estimation. We focus on humans, the most common "objects" in images.

Given a 2D image, we first exploit a deep network [158] to predict the 3D shape and pose of each person in the scene, as well as their relative poses to the camera. We use SMPL [225] as our representation since it allows us to reconstruct a complete human mesh from partial observations. Then we cast a ray through each pixel and record all the 3D points where the rays intersect with the human mesh via ray-plane intersection (see Fig.5-3-mid). Finally, we identify if any of those 3D points are visible in other images by 2D-3D association. If there is, we say the two pixel rays intersect in 3D and the corresponding two pixels are VCs. Specifically, we use DensePose [115] to associate each pixel with each point on the human mesh. If a ray hits the back of the mesh and DensePose tells us a pixel corresponds to the back, then these two

(a) SuperGlue (b) VCs

Figure 5-6: **Qualitative comparison.** Classic correspondence estimators fail when images have little overlap, since there are no co-visible 3D points. VCs can be found in both scenarios so long as the camera rays intersect. The color indicates epipolar error.

pixels are VCs. Fig. 5-3 illustrates the process, which we repeat for all images. We note that our formulation is generic and can be potentially applied to other objects so long as there exist proper shape priors and surface mapping. We show an example on cars in Sec. 5.4.

5.3.3 Generalized Bundle Adjustment (BA)

Once we establish virtual correspondences, the next step is to jointly refine the camera poses as well as the sparse 3D scene geometry. Similar to classic SfM, we initialize the camera poses using RANSAC with the five-point algorithm [121] in the loop[†]. But instead of employing classic correspondences, we use VCs.

Since VCs could correspond to different 3D points (see Fig. 5-2), traditional triangulation approach cannot recover both 3D points. We thus leverage the initial shape estimation (predicted by deep nets) to compute the ray-surface intersection and record the first hits for each VC. The 3D points are then registered into the global coordinate system using the estimated camera poses from the five-point algorithm.

Since the estimated structure (*i.e.* the sparse 3D points) and poses depend heavily on the predicted shape priors, they may be noisy. We further refine the estimates by minimizing the distance between reprojected points and VCs. Formally, let $(\mathbf{X}^{j_1}, \mathbf{X}^{j_2})$ be the j-th pair of reconstructed 3D points and $(\boldsymbol{p}_{i_1}, \boldsymbol{p}_{i_2})$ be the associated VC pair

[†]We assume the intrinsics are known or already estimated.

from camera i_1 and camera i_2. Denote $\alpha = (i_1, i_2, j_1, j_2)$ as a tuple of corresponding indices. Our goal is to minimize:

$$\begin{aligned} \min_{\mathbf{R}_i, \mathbf{t}_i, \mathbf{X}^{j_1}, \mathbf{X}^{j_2}} & \sum_{\alpha} \|\boldsymbol{p}_{i_1} - \pi_{i_1}(\mathbf{X}^{j_1})\|^2 + \|\boldsymbol{p}_{i_2} - \pi_{i_2}(\mathbf{X}^{j_2})\|^2 \\ \text{s.t.} \quad & \Big((\mathbf{X}^{j_1} - \mathbf{o}_{i_1}) \times (\mathbf{X}^{j_2} - \mathbf{o}_{i_2})\Big)^T (\mathbf{o}_{i_2} - \mathbf{o}_{i_1}) = 0, \end{aligned} \tag{5.2}$$

where $\pi_i(\mathbf{X}) \sim \mathbf{K}_i(\mathbf{R}_i\mathbf{X} + \mathbf{t}_i)$ is the perspective projection operator, and the constraint enforces the two camera rays to be co-planar such that epi-polar geometry holds.

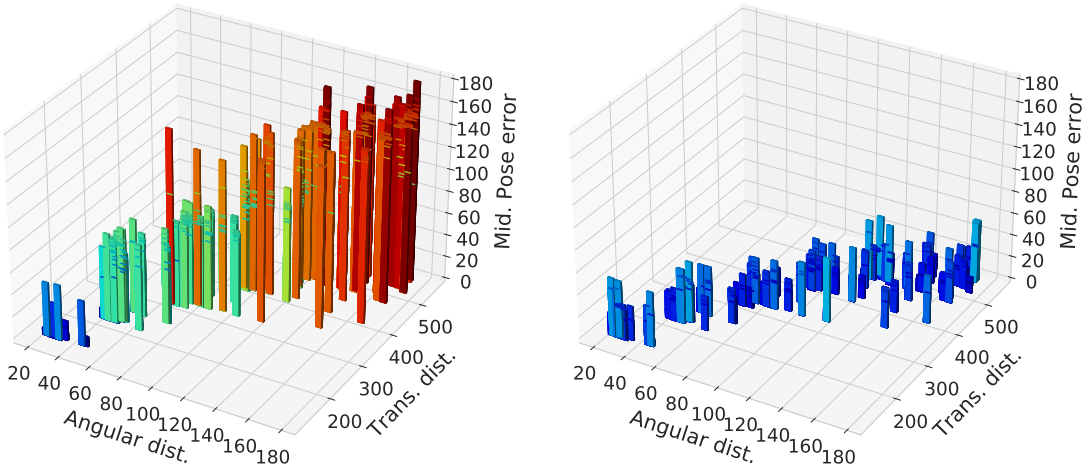
Using the constraint, we can further re-write one VC point as a function of the other:

$$\mathbf{X}^{j_2} = \mathbf{X}^{j_1} + a^j \cdot (\mathbf{X}^{j_1} - \mathbf{o}_{i_1}) + b^j \cdot (\mathbf{o}_{i_2} - \mathbf{o}_{i_1}). \tag{5.3}$$

The two free parameters a^j and b^j can be thought of as the "thickness" of the shape between the intersecting points. When both parameters become 0, the two 3D points merge into one, and VCs reduce to classic correspondences.

By replacing Eq. 5.3 into Eq. 5.2, we obtain an unconstrained minimization problem that is similar to, yet more generic than classic BA. Instead of refining a set of *co-visible 3D points*, we now adjust a bundle of *point tuples*. We, however, note that classic correspondences extracted with conventional methods such as SuperGlue [308] can still fit into this formulation by fixing $a^j = b^j = 0$. We use L-BFGS [260] to solve this non-linear least square problem. In practice, we treat Eq. 5.3 as a soft constraint since it works slightly better. We refer the readers to supp. material for more discussions.

Discussion: VCs can be combined with classic correspondences to improve the overall robustness and performance of 3D reconstruction systems (see Sec. 5.4). When the images barely overlap and few classic correspondences are available, the system can rely on VCs to recover the world and camera geometry. When the images do overlap, VCs can serve as additional visual cues and regularizers. VCs thus significantly expand the applicable setting of existing SfM systems.



(a) Classic SfM (b) Our SfM

Figure 5-7: **Pose error vs. ground-truth pose distance.** The median pose error in classic SfM (left) increases with increasing camera baseline, while the median pose error for our method (right) stays low regardless of viewpoint differences.

5.4 Experiments

In this section, we first evaluate the effectiveness of virtual correspondence and our 3D system on two challenging datasets. Then we comprehensively study the characteristics of our method. With the estimated camera poses, we further conduct two downstream tasks, namely scene reconstruction with multi-view stereo and novel view synthesis, in difficult extreme-view cases. Finally, to showcase that our method generalizes beyond human-based images, we demonstrate proof-of-concept results with cars.

5.4.1 Datasets

CMU Panoptic dataset: CMU Panoptic dataset [155, 156] is a large-scale, multi-view video dataset designed for human analysis. It provides ground-truth camera poses as well as person associations across views. The sequences were captured in a studio with (approximately) synchronized cameras widely spread across the dome, providing us a diverse set of viewpoints that are barely available in the real world (*e.g.*, cameras looking at a person from the top). We select 43 sequences from `pose`, `haggling`, and `dancing`. Each sequence contains 1∼3 people performing different actions. We divide the data into two splits. Each split comprises a set of unique sequences and cameras

without any overlap. Due to image quality, we only consider the videos captured by HD cameras. We sample a frame every five seconds to avoid similar human poses. We also run human detection on each sampled frame. If no person is present in the scene, we discard the frame. In total, we obtain 2955 image sets for each split, with each set containing 15-16 camera views. We refer the readers to the supp. material for more details.

Mannequin Challenge: Mannequin Challenge (MC) [199] is a dataset of internet video clips where the participants stay still in different poses, while the video-takers move freely in space and capture the event. These videos, by design, allow us to look at a static scene from various angles. We follow a similar pipeline as [199] to reconstruct the ground-truth camera trajectories and filter out snippets with small shifts in viewpoints or view directions. In the end, we obtain 18 video snippets where the cameras rotate by at least 90° within each sequence. To further increase pose diversity, we additionally collect 6 MC videos ourselves. Compared with the CMU dataset, the camera poses in MC videos are rather *generic* [101], yet the background scenes, which consist of both indoor and outdoor environments, are much more diverse. Finally, for each snippet, we compute the pose difference between each frame and the first frame. We sample a frame at every 20 percentile of the snippet and obtain ∼200 image pairs. All the images are treated as the test set.

5.4.2 Experimental Details

Metrics: Following previous work [409, 421, 34, 308], we employ the area under the cumulative error curve (AUC) to evaluate the recovered camera poses. We report the AUC at three different thresholds (15°, 30°, and 45°). The pose error is defined as the maximum of 1) the angular difference between predicted and GT rotation vectors; and 2) the angular difference between predicted and GT translation vectors. We report angular difference for translation since it can only be recovered up to a scaling factor [121]. As for 3D reconstruction, there is no standard protocol to compare point clouds directly produced by SfM systems because each SfM algorithm can choose which 3D

Pose estimation AUC (↑)	CMU Panoptic Studio			Mannequin Challenge		
Methods	@15°	@30°	@45°	@15°	@30°	@45°
SuperGlue [308]	10.02	16.74	19.36	26.38	34.85	39.10
LoFTR [334]	5.12	10.47	13.07	27.47	35.98	40.10
SIFT [226] + BA [314]	7.68	11.39	13.33	14.17	20.24	24.25
SuperPoint [72] + BA [314]	9.22	13.77	15.85	17.12	23.48	26.81
SuperGlue [308] + BA [314]	10.68	16.57	18.92	26.24	35.12	39.46
LoFTR [334] + BA [314]	8.35	14.52	17.01	27.51	36.32	40.55
Deep regression [179]	14.36	18.60	23.18	4.61	11.23	16.44
Deep optimization [30, 163]	7.88	27.17	42.42	15.38	47.08	63.67
Our SfM	**18.21**	**46.05**	**62.08**	**36.24**	**61.38**	**73.20**

Table 5.1: **Two frame relative pose estimation** on the CMU dataset and the MC dataset. First two rows perform five-point algorithm to derive camera poses. BA = Bundle Adjustment.

points to reconstruct. Furthermore, there is no ground-truth shape for both datasets. We thus follow [157] to compute the silhouette accuracy between the rendered mask and the 2D segmentation mask.

Baselines: We compare our method against a wide range of relative pose estimation methods. For traditional matching-based methods, we first detect the key points and extract their corresponding features with SIFT [226] or SuperPoint [72]. We then establish classic correspondences with either nearest neighbour matching with ratio test [226] or SuperGlue (SG) [308]. We also compare with LoFTR [334]. We further use RANSAC [97] coupled with the five-point algorithm to filter outliers. We then incrementally recover and bundle adjust the image poses with COLMAP [314]. Alternatively, if there are only two views, we also perform pose estimation with the five-point algorithm and essential matrix decomposition. Next, for deep regression methods, we employ a state-of-the-art pose estimation network [89] to predict the relative camera pose between an image pair. Finally, we compare against a deep optimization approach that estimates camera poses by aligning 3D shapes. The baseline is inspired by the state-of-the-art indoor extreme pose estimation method [279] and can be seen as a variant for humans. Specifically, we utilize the latest EFT-Net [158] to reconstruct 3D human models and align them with ICP [30]. To avoid local minima,

Initialization		BA		Pose estimation AUC		
SG	VCs	SG	VCs	@15°	@30°	@45°
✓	-	-	-	10.02	16.74	19.36
✓	✓	-	-	10.29	31.27	48.96
✓	-	✓	-	10.68	16.57	18.92
-	✓	-	✓	15.89	43.92	60.38
✓	✓	✓	✓	**18.21**	**46.05**	**62.08**

Table 5.2: **Ablation study** on the CMU dataset. SG refers to SuperGlue [308], a deep learning based classic correspondence.

we first register the shapes based on their canonical coordinates. Next, we associate each part of the shape based on its semantics. We further prune out the limbs and exploit only torso and head during matching since these two parts are more robust in practice. These strategies drastically improve the performance of this baseline.

Implementation details: Our 3D system considers both classic correspondences and VCs. We exploit SuperGlue [308] to estimate classic correspondences and ReID-Net [434] to match a person across multiple viewpoints. For the deep regression baseline, we train and validate on the training split of CMU dataset. For the rest of the learning based approaches, *including our method*, we adopt the pre-trained weights provided by the authors and conduct inference only.

5.4.3 Experimental Results

CMU Panoptic Studio: As shown in Tab. 5.1(left), our SfM outperforms all baselines at all thresholds in the two-frame pose estimation task. SuperGlue [308] ranks second when the pose error threshold is low, but deep optimization [30, 162] surpasses it when the threshold increases. This is expected since matching-based approaches can produce accurate estimation when classic correspondences are available, yet fail catastrophically when the viewpoints are very different. Deep optimization, in contrast, is not as accurate when the view difference is small, but has fewer fatal failures. Our approach, which exploits both classic and virtual correspondences, does not suffer from either catastrophic wide-baseline failures or inaccurate narrow-baseline

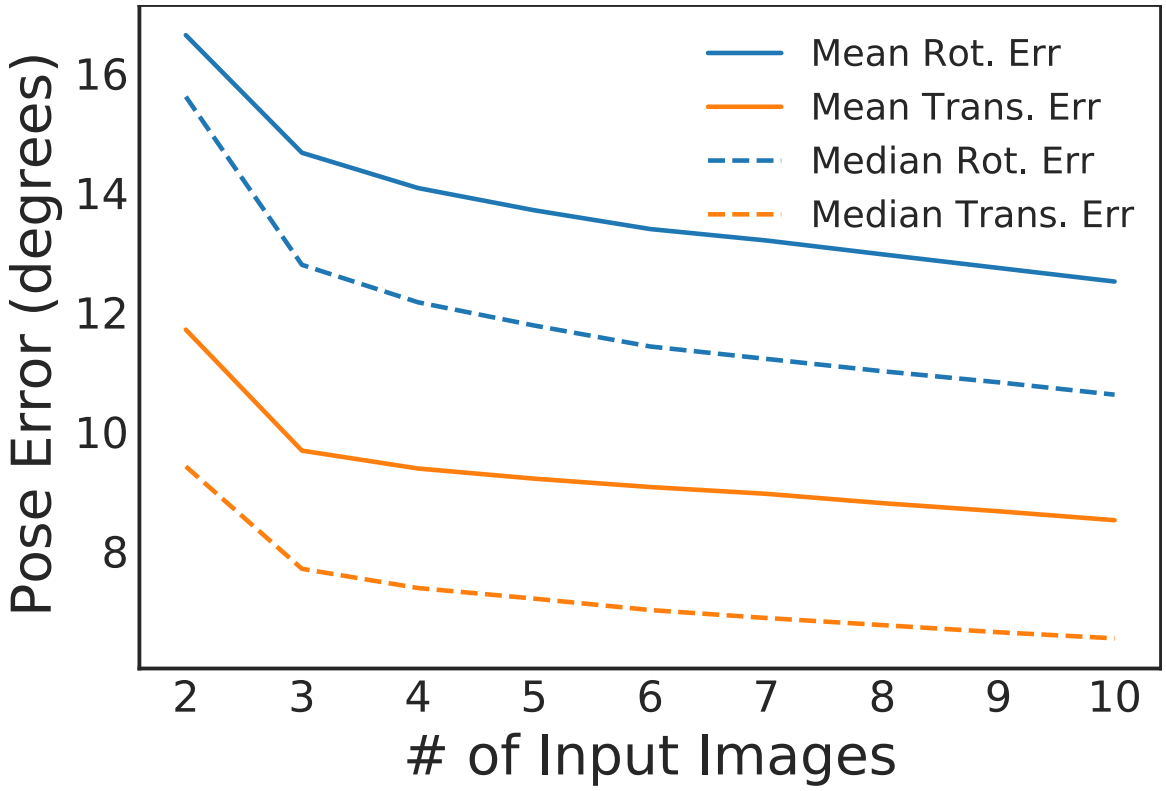


Figure 5-8: **Error vs. # of images.**

matching.

Our SfM has a median error of 15.7° and the pose error at the 80th percentile is less than 24°. In contrast, the median error of deep optimization is 23.5° and the pose error at the 80th percentile is 44°. Compared to EFT-Net, we improve the silhouette accuracy from 74% to 81%.

We also investigate how our SfM scales with more input images. Following COLMAP [314], we start from an image pair and then incrementally register new images. As shown in Fig. 5-8, the pose error reduces as more images are added. The reduction is most significant when registering the third image. We hypothesize this is because the third image greatly increases the overlap among the images, providing more reliable classic correspondences during bundle adjustment. We also compare our approach with the classic SfM methods. Our AUC continuously outperforms the baselines at all thresholds (*e.g.*, @15°: 28.4 vs 17.6). We refer the readers to the supp. material for full ablation table, cumulative error plots, and detailed performance

Figure 5-9: **Reconstructed mesh** using our method + multi-view stereo for two non-overlapping video sequences.

of all methods with respect to the input images.

Mannequin Challenge: As shown in Tab. 5.1(right), our method outperforms all baselines at all thresholds. Despite more diverse scenes, our AUC on the MC dataset is higher than that of the CMU dataset. We hypothesize this is because the viewpoint changes in the MC dataset are less significant than the CMU dataset, due to how the dataset was collected.

Qualitative results: We showcase our results on a two-view MC image pair, and a five-view CMU image set in Fig. 5-4. Our testing scenarios are typically very challenging, with large view variations and small proportion of co-visible regions. Nevertheless, our proposed SfM framework is able to recover both relative poses as well as the parametric human shape accurately.

5.4.4 Analysis

Ablation study: To gain more insights into the contribution of each component, we evaluate our method with different configurations on the CMU dataset. As shown in Tab. 5.2, by simply exploiting VCs during initialization, our method surpasses classic SfM in terms of AUC at large thresholds. Additionally, the ablation study shows that bundle adjustment is critical for VCs. We conjecture this is because VCs are constructed from initial shape priors, which are noisy. By bundle adjusting the line segments, we are essentially conducting maximum likelihood estimation [121]

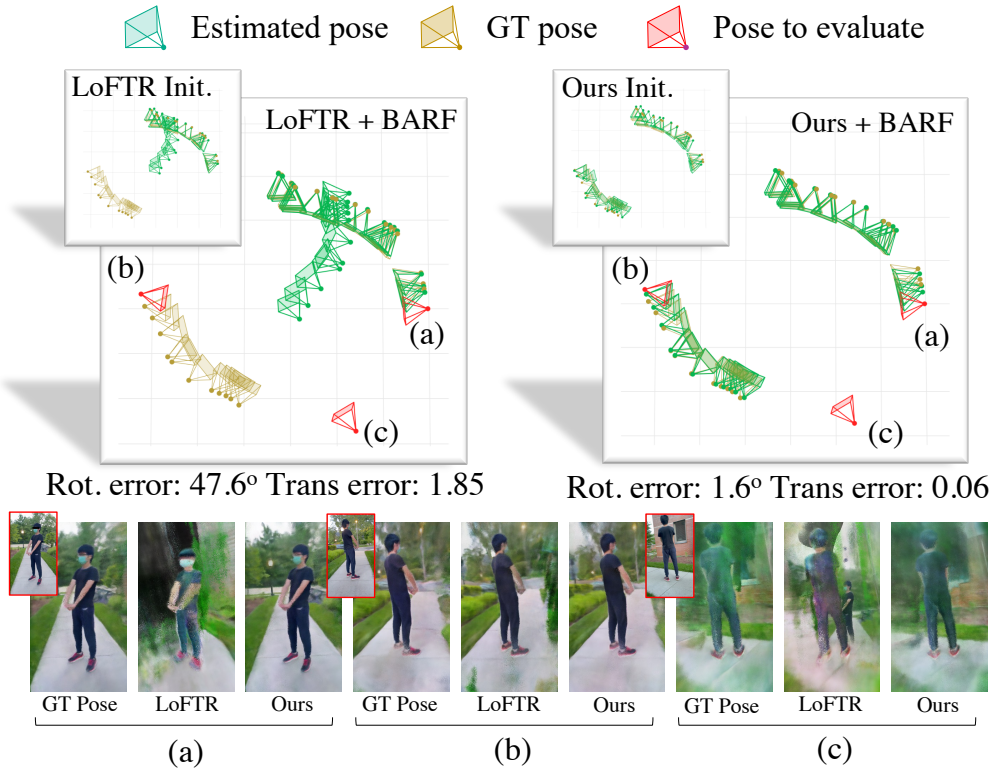


Figure 5-10: **Novel view synthesis:** (top left) camera poses initialized with LoFTR and refined by BARF; (top right) camera poses initialized with our method and refined by BARF; (bottom) images synthesized at novel views by BARF initialized with GT pose, LoFTR and our framework respectively.

under Gaussian noise assumption on VC re-projection errors, which mitigates errors introduced by inaccurate VC pairs.

Effects of viewpoint changes: We use the MC dataset to illustrate how classic and virtual correspondences evolve with viewpoint changes and how it affects pose estimation. In general, the ground-truth camera pose difference is proportional to the video frame index distance. For each video, we compute the classic correspondences and VCs between all frames and the first frame, and then estimate the relative camera poses based on them. Since the number of classic correspondences decreases drastically when the viewpoint changes, classic SfM fails. In contrast, our SfM framework incorporates both classic correspondences and VCs to avoid failures. Fig. 5-5 shows an example of how our system produces decent estimation across all distances. We also

showcase a “discrete" evaluation on the CMU dataset in Fig. 5-7. The pose error of classic SfM methods increases significantly with respect to the ground-truth camera pose distances (the diagonal direction), while our SfM performs consistently across all settings.

Reliability of human parts: We compute the histogram over all VCs on both datasets. Around half of the VCs lie on human torso, and around 12% of VCs are derived from the human head. The remaining VCs uniformly spread across the whole body. Unlike the deep optimization baseline, we do not encode any prior knowledge into our system, yet our approach is able to automatically discover that human torso is the most reliable parts within the predicted 3D shapes. Fig. 5-6 shows a subset of VCs selected by our SfM system.

Generalization to in-the-wild images: Our approach can be applied to real-world image collections without bells and whistles. We test our system on a pair of movie frames and two pairs of sports photos in Fig. 5-1. Even though the cameras are far apart and the images are slightly asynchronous, our system still produces reasonable estimates. More results on classic movies and sports events can be found in supp. materials.

Limitations: Our approach relies heavily on the predicted shape priors. While we can handle noisy predictions by pruning out the outlier VCs during geometric verification, if the initial estimation is completely wrong (which our system can detect by comparing silhouette consistency, DensePose consistency, *etc.*), we will not be able to construct VCs. Additionally, similar to classic SfM algorithms, we assume the scene is static. While we can tolerate slight movements (see Fig. 5-1), it fails when human poses change significantly.

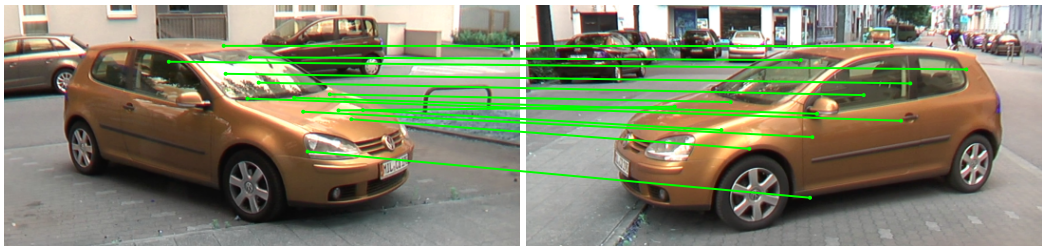


Figure 5-11: **Virtual correspondences from cars.**

5.4.5 Applications

Scene reconstruction with multi-view stereo (MVS): We first show how VCs enable coherent multi-view 3D reconstruction from non-overlapping videos. This type of capture is common in practice, yet most SfM and MVS systems can only handle each sequence individually, resulting in two disjoint 3D reconstructions. Our VCs, in contrast, are able to recover relative poses even from non-overlapping images, which allows us to obtain a single, coherent MVS point cloud to unlock further geometry processing such as mesh reconstruction. We use RealityCapture [2] to reconstruct the 3D scene by initializing the camera poses with our estimation. The resulting high-quality meshes suggest that the recovered camera poses and the extracted point clouds are accurate (see Fig. 5-9). In contrast, both RealityCapture's built-in 3D reconstruction pipeline and COLMAP [314] fail due to non-overlapping viewpoints.

Novel view synthesis: We further demonstrate the effectiveness of our approach in extreme-view scenarios through the task of novel view synthesis, which relies heavily on input poses. In particular, we adopt BARF [210], an approach that can learn a neural radiance field [246] and refine camera poses simultaneously. We again use two non-overlapping video sequences. We initialize BARF with the poses recovered by our method and LoFTR [334] respectively. Our estimated poses, which are already fairly accurate, are further refined through the course of BARF training (see Fig. 5-10). In contrast, LoFTR [334] fails to estimate the relative camera poses among the two sequences correctly (see the green cameras) and the resulting BARF training gets stuck in local minima. We also evaluate the learned radiance field with novel, extrapolated poses. Our view syntheses results are comparable to those trained with

GT poses. As expected, the quality degrades when the evaluation pose deviates too far from the training poses, especially for the background scene that is unseen in the training videos. However, we can still see a person standing on the pavement and observe the structure of the scene. On the other hand, due to the incorrect LoFTR poses, the baseline fails to produce realistic results.

Extending VCs to other objects: As a proof-of-concept, we exploit canonical 3D deformable mapping [261] as shape priors and adapt our method to cars. As shown in Fig. 5-11, we are able to estimate VCs and recover relative poses effectively (pose error: 16°) even from extreme viewpoints. We refer the readers to supp. material for more details.

5.5 Conclusion

We introduced a novel concept called virtual correspondences – a pair of image points whose camera rays intersect in 3D. Unlike classic correspondences, virtual correspondences do not need to describe the same, co-visible 3D points. Thus, VCs are not constrained by visual or semantic similarities, making it possible to match images with little or no overlap. We proposed a method to extract virtual correspondences based on prior knowledge of foreground objects in the image, and integrate with existing 3D frameworks. Our experiments on two challenging human-based datasets show that virtual correspondences are critical towards successful camera pose estimation and downstream multi-view stereo and novel view synthesis in extreme-view scenarios.

Social impact: Our method alleviates the need to capture dense views for camera pose estimation and 3D reconstruction, and has the potential to reduce storage and computational costs. Unfortunately, it could also be exploited by surveillance and may raise privacy concerns as 3D reconstruction from few images becomes more accessible.

Chapter 6

Extreme 3D Modeling with Neural Radiance Fields

Structure *from* Duplicates:
Inverse Graphics from a Pile of Objects
Tianhang Chen*, Wei-Chiu Ma*, Kaiyu Guan, Antonio Torralba, Shenlong Wang;
arXiv 2023.

We have shown that a high-level understanding of objects can enhance our reasoning about geometric relationships within a scene. Next, we take a step further and explore if we can also recover the underlying physical properties, such as albedo, roughness, metallicity, and illumination, from an extreme setup. We focus on the setting where multiple (nearly) identical instances are present within a scene. By establishing a duality between multiple copies of an object in a single image and multiple views of a single object, we are able to resolve the ambiguities in 3D and effectively recover both the geometric and physical properties of interest.

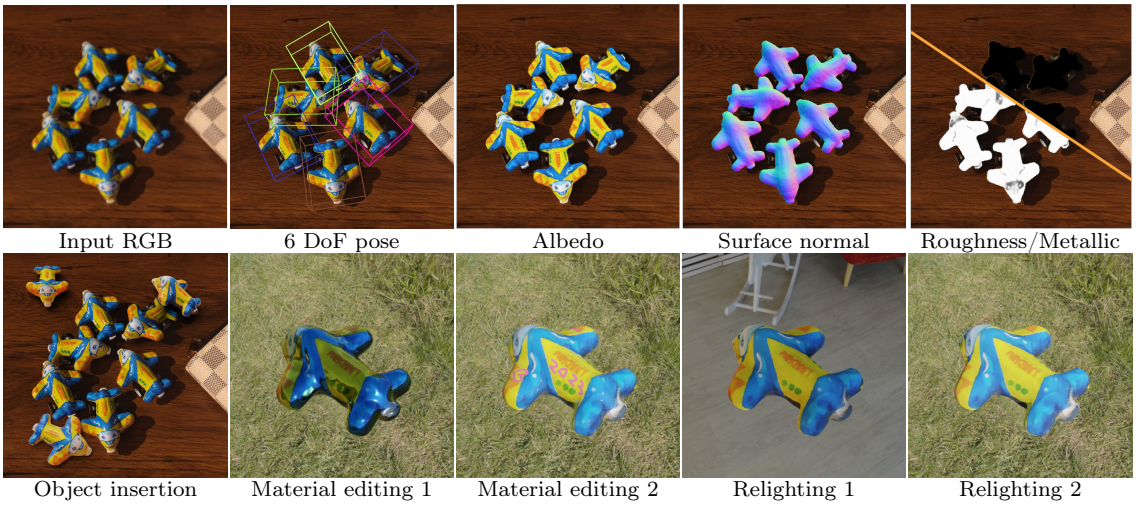


Figure 6-1: **Structure from duplicates (SfD)** is a novel inverse graphics framework that reconstructs geometry, material, and illumination from a single image containing multiple identical objects.

6.1 Introduction

Given a single/set of image(s), the goal of inverse rendering is to recover the underlying geometry, material, and lighting of the scene. The task is of paramount interest to many applications in computer vision, graphics, and robotics and has drawn extensive attention across the communities over the past few years [430, 249, 123, 246].

Since the problem is ill-posed, prevailing inverse rendering approaches often leverage multi-view observations to constrain the solution space. While these methods have achieved state-of-the-art performance, in practice, it is sometimes difficult, or even impossible, to obtain those densely captured images. To overcome the reliance on multi-view information, researchers have sought to incorporate various structural priors, either data-driven or handcrafted, into the models [55, 204]. By utilizing the regularizations, these approaches are able to approximate the intrinsic properties (*e.g.*, material) and extrinsic factors (*e.g.*, illumination) even from one single image. Unfortunately, the estimations may be biased due to the priors imposed. This makes one ponder: is it possible that we take the best of both worlds? Can we extract multi-view information from a single image under certain circumstances?

Fortunately the answer is yes. Our world is full of repetitive objects and struc-

tures. Repetitive patterns in single images can help us extract and utilize multi-view information. For instance, when we enter an auditorium, we often see many identical chairs facing slightly different directions. Similarly, when we go to a supermarket, we may observe multiple nearly-identical apples piled on the fruit stand. Although we may not see *the exact same object* from multiple viewpoints in just one glance, we do see many of the "identical twins" from various angles, which is equivalent to multi-view observations and even more (see Sec. 6.3 for more details). Therefore, the goal of this chapter is to develop a computational model that can effectively infer the underlying 3D representations from a single image by harnessing the repetitive structures of the world.

With these motivations in mind, we present Structure from Duplicates (SfD), a novel inverse rendering model that is capable of recovering high-quality geometry, material, and lighting of the objects from a single image. SfD builds upon insights from structure from motion (SfM) as well as recent advances on neural fields. At its core lies two key modules: (i) a *in-plane rotation robust pose estimation module*, and (ii) a *geometric reconstruction module*. Given an image of a scene with duplicate objects, we first exploit the pose estimation module to estimate the relative 6 DoF poses of the objects. Then, based on the estimated poses, we align the objects and create multiple "virtual cameras." This allows us to effectively map the problem from a single-view multi-object setup to a multi-view single-object setting (see Fig. 6-3). Finally, once we obtain multi-view observations, we can leverage the geometric module to recover the underlying intrinsic and extrinsic properties of the scene. Importantly, SfD can be easily extended to multi-image setup. It can also be seen as a superset of existing NeRF models, where the model will reduce to NeRF when there is only one single object in the scene.

We validate the efficacy of our model on a new dataset called **Dup**, which contains synthetic and real-world samples of duplicated objects since current multi-view datasets lack duplication samples. This allows us to benchmark inverse rendering performance under single-view or multi-view settings. Following previous work [364, 408, 249, 424, 430], we evaluate rendering, relighting and texture quality with MSE, PSNR, SSIM,

Figure 6-2: **Repetitions in the visual world.** Our physical world is full of identical objects (*e.g.*, cans of coke, cars of the same model, chairs in a classroom). These duplicates, when seen together, provide additional and strong cues for us to effectively reason about 3D.

LPIPS [427], geometry with Chamfer Distance (CD), and environment light with MSE. Experimental results suggest that 1) our method produces more realistic material texture than the existing multi-view inverse rendering model when using the same number of training views; 2) even only relying on a single-view input, our approach can still recover comparable or superior materials and geometry compared to baselines that utilize multi-view images for supervision.

6.2 Related Work

Inverse rendering: The task of inverse rendering can be dated back to more than half a century ago [185, 136, 137, 22]. The goal is to factorize the appearance of an object or a scene in the observed image(s) into underlying geometry, material properties, and lighting conditions [310, 238, 411]. Since the problem is severely under-constrained, previous work mainly focused on controlled settings or simplifications of the problem [112, 124, 19]. For instance, they either assume the reflectance of an object is spatially invariant [424], presume the lighting conditions are known [329], assume the materials are lambertian [436, 230], or presume the proxy geometry is available [309, 188, 77, 183]. More recently, with the help of machine learning, in particular deep learning, researchers have gradually moved towards more challenging in-the-wild settings [432, 249]. By pre-training on a large amount of synthetic yet realistic data [204] or baking inductive biases into the modeling pipeline [57, 428], these

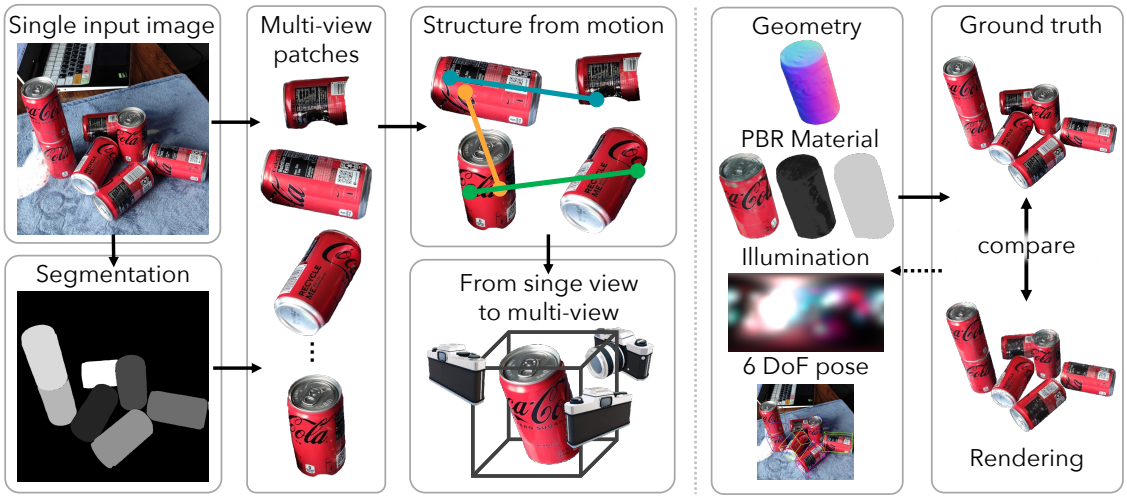


Figure 6-3: **Method overview: (Left)** SfD begins by identifying multiple instances of an object within an image, and then jointly estimates the 6DoF pose for all instances. **(Right)** An inverse graphics pipeline is subsequently employed to reason about the shape, material of the object, and the environment light, while adhering to the shared geometry and material constraint across instances.

approaches can better tackle unconstrained real-world scenarios (*e.g.*, unknown lighting conditions) and recover the underly physical properties more effectively [249, 425]. For example, through properly modeling the indirect illumination and the visibility of direct illumination, Zhang *et al.* [430] is able to recover interreflection- and shadow-free SVBRDF materials (*e.g.*, albedo, roughness). Through disentangling complex geometry and materials from lighting effects, Wang *et al.* [379] can faithfully relight and manipulate a large outdoor urban scene. Our work builds upon recent advances in neural inverse rendering. Yet instead of grounding the underlying physical properties through multi-view observations as in prior work, we focus on the single image setup and capitalize on the duplicate objects in the scene for regularization. The repetitive structure not only allows us to ground the geometry, but also provide additional cues on higher-order lighting effects (*e.g.*, cast shadows). As we will show in the experimental section, we can recover the geometry, materials, and lighting much more effectively even when comparing to multi-view observations.

3D Reconstruction: Recovering the spatial layout of the cameras and the geometry of the scene from a single or a collection of images is a longstanding challenge

in computer vision. It is also the cornerstone for various downstream applications in computer graphics and robotics such as inverse rendering [238, 364, 379], 3D editing [221, 399], navigation [231, 418], and robot manipulation [145, 213]. Prevailing 3D reconstruction systems, such as structure from motion (SfM), primarily rely on multi-view geometry to estimate the 3D structure of a scene [224, 122, 314]. While achieving significant successes, they rely on densely captured images, limiting their flexibility and practical use cases. Single image 3D reconstruction, on the other hand, aims to recover metric 3D information from a monocular image [87, 56, 286, 287]. Since the problem is inherently ill-posed and lacks the ability to leverage multi-view geometry for regularization, these methods have to resort to (learned) structural priors to resolve the ambiguities. While they offer greater flexibility, their estimations may inherit biases from the training data. In this chapter, we demonstrate that, under certain conditions, it is possible to incorporate multi-view geometry into a single image reconstruction system. Specifically, we leverage repetitive objects within the scene to anchor the underlying 3D structure. By treating each of these duplicates as an observation from different viewpoints, we can achieve highly accurate metric 3D reconstruction from a single image.

Repetitions: Repetitive structures and patterns are ubiquitous in natural images. They play important roles in addressing numerous computer vision problems. For instance, a single natural image often contains substantial redundant patches [441]. The recurrence of small image patches allows one to learn a powerful prior which can later be utilized for various tasks such as super-resolution [107, 141], image deblurring [245], image denoising [88], and texture synthesis [85]. Moving beyond patches, repetitive primitives or objects within the scene also provide informative cues about their intrinsic properties [195, 402]. By sharing or regularizing their underlying representation, one can more effectively constrain and reconstruct their 3D geometry [139, 96], as well as enable various powerful image/shape manipulation operations [196, 363]. In this work, we further push the boundary and attempt to recover not just the geometry, but also the materials (*e.g.*, albedo, roughness), visibilities, and

Multi-view	Albedo PSNR ↑	Roughness MSE ↓	Relighting PSNR ↑	Env. Light MSE ↓	Geometry CD ↓
PhySG	16.233	0.087	21.323	0.054	0.024
Nv-DiffRec	16.123	0.116	17.418	0.168	0.268
InvRender	16.984	0.084	22.224	0.067	0.024
Ours	**21.961**	**0.026**	**25.486**	**0.029**	**0.011**

Table 6.1: **Multi-view inverse rendering on synthetic data**. Both our model and the baseline are trained on multi-view images. Our model is significantly better than baseline in terms of geometry and PBR texture.

Single-view	Albedo PSNR ↑	Roughness MSE ↓	Relighting PSNR ↑	Env. Light MSE ↓	Geometry CD ↓
PhySG*	14.977	0.255	18.504	0.082	**0.033**
Nv-DiffRec*	14.021	0.165	17.214	0.067	0.050
InvRender*	14.724	0.247	17.998	0.082	0.033
Ours	**17.629**	**0.062**	**21.374**	**0.052**	0.034

Table 6.2: **Single-view inverse rendering on synthetic data.** While our model is trained on a *single-view image*, the baselines $*$ are trained on 10 *multi-view images* of the same scene.

lighting conditions of the objects. Perhaps closest to our work is [431]. Developed independently and concurrently, Zhang *et al.* build a generative model that aims to capture object intrinsics from a single image with multiple similar/same instances. However, there exist several key differences: 1) we explicitly recover metric-accurate camera poses using multi-geometry, whereas Zhang et al. learn this indirectly through a GAN-loss; 2) we parameterize and reason realistic PBR material and environmental light; 3) we handle arbitrary poses, instead of needing to incorporate a prior pose distribution.

6.3 Structure from Duplicates

In this chapter, we seek to devise a method that can precisely reconstruct the geometry, material properties, and lighting conditions of an object from *a single image containing duplicates of it*. We build our model based on the observation that repetitive objects in the scene often have different poses and interact with the environment (*e.g.*, illumination) differently. This allows one to extract rich *multi-view* information even

from one single view and enables one to recover the underlying physical properties of the objects effectively.

We start by introducing a method for extracting the "multi-view" information from duplicate objects. Then we discuss how to exploit recent advances in neural inverse rendering to disentangle both the object intrinsics and environment extrinsics from the appearance. Finally, we describe our learning procedure and design choices.

6.3.1 Collaborative 6-DoF pose estimation

As we have alluded to above, a single image with multiple duplicate objects contains rich multi-view information. It can help us ground the underlying geometry and materials of the objects, and understand the lighting condition of the scene.

Our key insight is that the image can be seen as *a collection of multi-view images stitching together*. By cropping out each object, we can essentially transform the single image into a set of multi-view images of the object from various viewpoints. One can then leverage structure from motion (SfM) [316] to estimate the relative poses among the multi-view images, thereby aggregating the information needed for inverse rendering. Notably, the estimated camera poses can be inverted to recover the 6 DoF poses of the duplicate objects. As we will elaborate in Sec. 6.3.2, this empowers us to more effectively model the extrinsic lighting effect (which mainly depends on world coordinate) as well as to properly position objects in perspective and reconstruct the exact same scene.

To be more formal, let $\mathcal{I} \in \mathbb{R}^{H\times W\times 3}$ be an image with N duplicate objects. Let $\{\mathcal{I}_i^{\text{obj}}\}_{i=1}^N \in \mathbb{R}^{w\times h\times 3}$ be the corresponding object image patches. We first leverage SfM [316, 307] to estimate the camera poses of the multi-view cropped images* $\{\boldsymbol{\xi}_i \in \mathbb{SE}(3)\}_{i=1}^N$:

$$\boldsymbol{\xi}_1^{\text{cam}}, \boldsymbol{\xi}_2^{\text{cam}}, ..., \boldsymbol{\xi}_N^{\text{cam}} = f^{\text{SfM}}(\mathcal{I}_1^{\text{obj}}, \mathcal{I}_2^{\text{obj}}, ..., \mathcal{I}_K^{\text{obj}}). \tag{6.1}$$

*In practice, the cropping operation will change the intrinsic matrix of the original camera during implementation. For simplicity, we assume the intrinsics are properly handled here.

Next, since there is only one real camera in practice, we can simply align the N virtual cameras $\{\boldsymbol{\xi}_i\}_{i=1}^{N}$ to obtain the 6 DoF poses of the duplicate objects. Without loss of generality and for simplicity, we align all the cameras to a reference coordinate $\boldsymbol{\xi}^{\text{ref}}$. The 6 DoF poses of the duplicate objects thus become $\boldsymbol{\xi}_i^{\text{obj}} = \boldsymbol{\xi}^{\text{ref}} \circ (\boldsymbol{\xi}_i^{\text{cam}})^{-1}$, where $\circ$ is matrix multiplication for pose composition.

In practice, we first employ a state-of-the-art panoptic segmentation model [61] to segment all objects in the scene. Then we fit a predefined bounding box to each object and crop it. Lastly, we run COLMAP [316] to estimate the 6 DoF virtual camera poses, which in turn provides us with the 6 DoF object poses. Fig. 6-3(left) depicts our collaborative 6-DoF pose estimation process.

Caveats of random object poses: Unfortunately, naively feeding these object patches into SfM would often leads to failure, as little correspondence can be found. This is due to the fact that state-of-the-art correspondence estimators [307] are trained on Internet vision data, where objects are primarily upright. The poses of the duplicate objects in our case, however, vary significantly. Moreover, the objects are often viewed from accidental viewpoints [101]. Existing estimators thus struggle to match effectively across such extreme views.

Rotation-aware data augmentation: Fortunately, the scene contains numerous duplicate objects. While estimating correspondences reliably across arbitrary instances may not always be possible, there are certain objects whose viewpoints become significantly similar after in-plane rotation. Hence, we have developed an in-plane rotation-aware data augmentation for correspondence estimation.

Specifically, when estimating correspondences between a pair of images, we don't match the images directly. Instead, we gradually rotate one image and then perform the match. The number of correspondences at each rotation is recorded and then smoothed using a running average. We take the `argmax` to determine the optimal rotation angle. Finally, we rotate the correspondences from the best match back to the original pixel coordinates. As we will demonstrate in Sec. 6.4, this straightforward data

# of instances	Albedo PSNR ↑	Roughness MSE ↓	Env Light MSE ↓	Geometry CD ↓
2	15.024	0.076	0.069	0.536
4	15.874	0.087	0.082	0.225
6	16.713	0.067	0.067	0.161
8	19.440	0.129	0.082	0.093
10	**22.720**	**0.057**	**0.052**	**0.024**

Table 6.3: **Performance vs. number of duplicates.**

	Rendering		
	PSNR ↑	SSIM ↑	LPIPS ↓
PhySG*	**20.624**	0.641	0.263
Nv-DiffRec*	18.818	0.569	0.282
InvRender*	20.665	0.639	0.262
Ours	20.326	**0.660**	**0.192**

Table 6.4: **Single-view inverse rendering on real-world data.** $*$ indicates that the baselines are trained on multi-view observations.

augmentation strategy significantly improves the accuracy of 6 DoF pose estimation. In practice, we rotate the image by 2° per step. All the rotated images are batched together, enabling us to match the image pairs in a single forward pass.

6.3.2 Joint shape, material, and illumination estimation

Suppose now we have the 6 DoF poses of the objects $\{\boldsymbol{\xi}_i^{\text{obj}}\}_{i=1}^N$. The next step is to aggregate the information across duplicate objects to recover the *intrinsics properties of the objects* (*e.g.*, geometry, materials) and *the extrinsic factors of the world* (*e.g.*, illumination). We aim to reproduce these attributes as faithfully as possible, so that the resulting estimations can be utilized for downstream tasks such as relighting and material editing. Since the task is under-constrained and joint estimation often leads to suboptimal results, we follow prior art [430, 379] and adopt a stage-wise procedure.

Geometry reconstruction: We start by reconstructing the geometry of the objects. In line with NeuS [364], we represent the object surfaces as the zero level set of a signed distance function (SDF). We parameterize the SDF with a multi-layer perceptron (MLP) $S : \mathbf{x}^{\text{obj}} \mapsto s$ that maps a 3D point under object coordinate $\mathbf{x}^{\text{obj}} \in \mathbb{R}^3$ to a

signed distance value $s \in \mathbb{R}$. Different from NeuS [364], we model the geometry of objects in local object space. This allows us to guarantee shape consistency across instances by design. We can also obtain the surface normal by taking the gradient of the SDF: $\mathbf{n}(\mathbf{x}^{\text{obj}}) = \nabla_{\mathbf{x}^{\text{obj}}} S$. To learn the geometry from multi-view images, we additionally adopt an *auxiliary* appearance MLP $C : \{\mathbf{x}, \mathbf{x}^{\text{obj}}, \mathbf{n}, \mathbf{n}^{\text{obj}}, \boldsymbol{d}, \boldsymbol{d}^{\text{obj}}\} \mapsto \mathbf{c}$ that takes as input a 3D point $\mathbf{x}, \mathbf{x}^{\text{obj}}$, surface normal $\mathbf{n}, \mathbf{n}^{\text{obj}}$, and view direction $\boldsymbol{d}, \boldsymbol{d}^{\text{obj}}$ under both coordinate systems and outputs the color $\mathbf{c} \in \mathbb{R}^3$. The input from world coordinate system helps the MLP to handle the appearance inconsistencies across instances caused by lighting or occlusion. We tied the weights of the early layers of C to those of S so that the gradient from color can be propagated to geometry. We determine which object coordinate to use based on the object the ray hits. This information can be derived either from the instance segmentation or another allocation MLP $A : \{\mathbf{x}, \boldsymbol{d}\} \mapsto q$, where $q \in \mathbb{N}$ is the instance index. After we obtain the geometry MLP S, we discard the auxiliary appearance MLP C. As we will discuss in the next paragraph, we model the object appearance using physics-based rendering (PBR) materials so that it can handle complex real-world lighting scenarios.

Material and illumination model: Now we have recovered the geometry of the objects, the next step is to estimate the environment light of the scene as well as the material of the object. We assume all lights come from an infinitely faraway sphere and only consider direct illumination. Therefore, the illumination emitted to a 3D point in a certain direction is determined solely by the incident light direction $\boldsymbol{w}_i$ and is independent of the point's position. Similar to [424, 430], we approximate the environment light with $M = 128$ Spherical Gaussians (SGs):

$$L_i(\omega_i) = \sum_{k=1}^{M} \boldsymbol{\mu}_k e^{\lambda_k(\boldsymbol{w}_i \cdot \boldsymbol{\phi}_k - 1)}, \tag{6.2}$$

where $\lambda \in \mathbb{R}^+$ is the lobe sharpness, $\boldsymbol{\mu}$ is the lobe amplitude, and $\boldsymbol{\phi}$ is the lobe axis. This allows us to effectively represent the illumination and compute the rendering equation (Eq. 6.3) in closed-form.

As for object material, we adopt the simplified Disney BRDF formulation [38, 164] and parameterized it as a MLP $M : \mathbf{x}^{\text{obj}} \mapsto \{\mathbf{a}, r, m\}$. Here, $\mathbf{a} \in \mathbb{R}^3$ denotes albedo, $r \in [0, 1]$ corresponds to roughness, and $m \in [0, 1]$ signifies metallic. Additionally, inspired by InvRender [430], we incorporate a visibility MLP $V : (\mathbf{x}, \boldsymbol{w}_i) \mapsto v \in [0, 1]$ to approximate visibility for faster reference. We query only surface points, which can be derived from the geometry MLP S using volume rendering. The material MLP M also operates in object coordinates like S, which ensure material consistency across all instances. Moreover, the variations in lighting conditions between instances help us better disentangle the effects of lighting from the materials. We set the dielectrics Fresnel term to $F_0 = 0.02$ and the general Fresnel term to $\mathbf{F} = (1 - m)F_0 + m\mathbf{a}$ to be compatible with both metals and dielectrics.

Combining all of these components, we can generate high-quality images by integrating the visible incident lights from hemisphere and modeling the effects of BRDF [160]:

$$L_o(\boldsymbol{w}_o; \mathbf{x}) = \int_\Omega L_i(\boldsymbol{w}_i) f_r(\boldsymbol{w}_i, \boldsymbol{w}_o; \mathbf{x})(\boldsymbol{w}_i \cdot \mathbf{n}) d\boldsymbol{w}_i. \tag{6.3}$$

Here, $\boldsymbol{w}_i$ is the incident light direction, while $\boldsymbol{w}_o$ is the viewing direction. The BRDF function f_r can be derived from our PBR materials. We determine the visibility of an incident light either through sphere tracing or following Zhang *et al.* [430] to approximate it with a visibility MLP V.

6.3.3 Optimization

Optimizing shape, illumination, and material jointly from scratch is challenging. Taking inspiration from previous successful approaches [430, 424], we implement a multi-stage optimization pipeline. We progressively optimize the geometry first, then visibility, and finally, material and illumination.

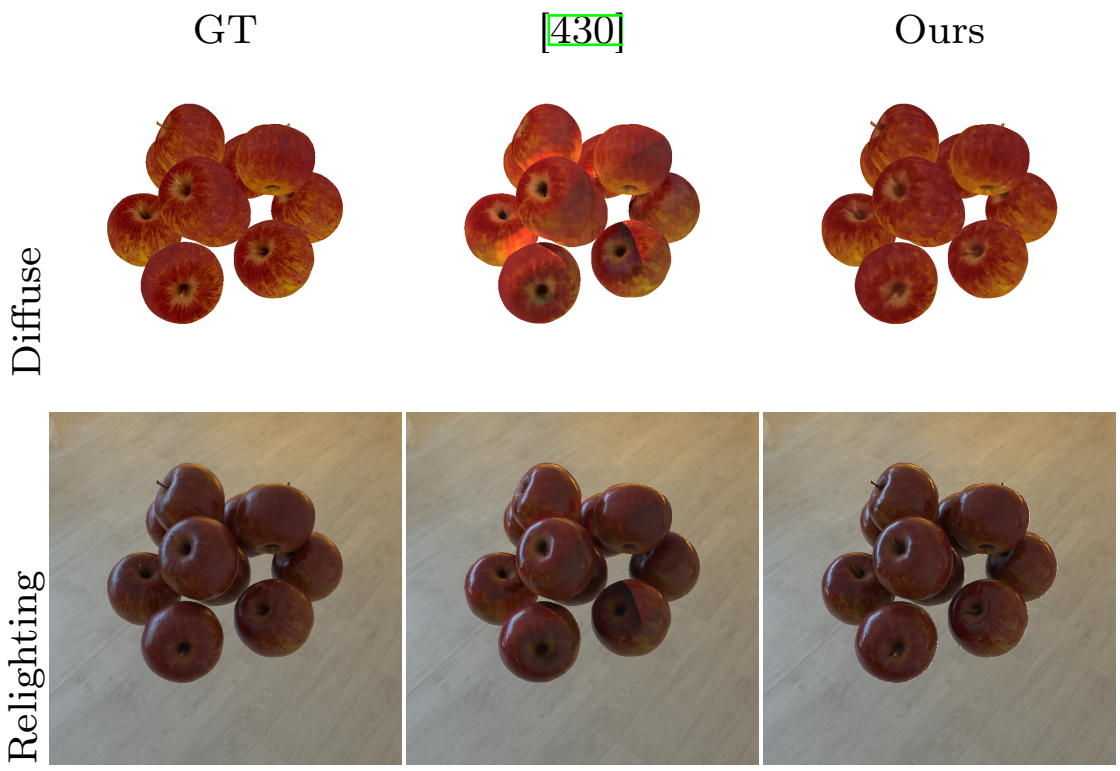


Figure 6-4: **Multi-view inverse rendering.**

Geometry optimization: We optimize the geometry model by minimizing the difference between rendered cues and observed cues.

$$\min_{S,C} E_{\text{color}} + E_{\text{reg}} + E_{\text{mask}} + E_{\text{normal}}, \tag{6.4}$$

where each term is defined as follows:

- The *color consistency term* E_{color} is a L1 color consistency loss between the rendered color $\mathbf{c}$ and the observed color $\hat{\mathbf{c}}$ for all pixel rays: $E_{\text{color}} = \sum_{\mathbf{r}} \|\mathbf{c}_{\mathbf{r}} - \mathbf{c}_{\mathbf{r}}\|_1$.
- The *normal consistency term* E_{normal} measures the rendered normal $\hat{\mathbf{n}}$ and a predicted normal $\hat{\mathbf{n}}$: $E_{\text{normal}} = \sum_{\mathbf{r}} \|1 - \hat{\mathbf{n}}_{\mathbf{r}}^{\mathrm{T}} \mathbf{n}_{\mathbf{r}}\|_1 + \|\hat{\mathbf{n}}_{\mathbf{r}} - \mathbf{n}_{\mathbf{r}}\|_1$. Our monocular predicted normal $\hat{\mathbf{n}}$ is obtained using a pretrained Omnidata model [86].
- The *mask consistency term* E_{mask} measures the discrepancy between the rendered mask $\mathbf{m}_{\mathbf{r}}$ and the observed mask $\hat{\mathbf{m}}_{\mathbf{r}}$, in terms of binary cross-entropy (BCE): $L_{\text{mask}} = \sum_{\mathbf{r}} \text{BCE}\left(\mathbf{m}_{\mathbf{r}}, \hat{\mathbf{m}}_{\mathbf{r}}\right)$.

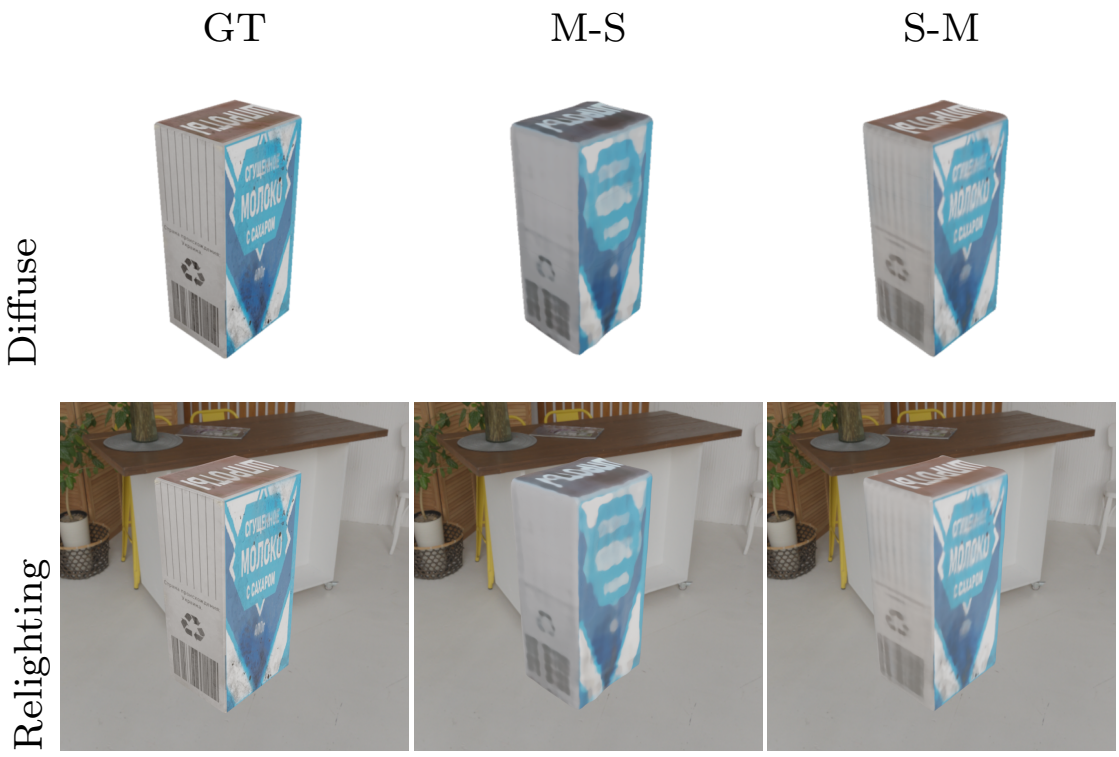


Figure 6-5: **Multi-view single object (M-S) vs single-view multi-objects (S-M).**

- Finally, inspired by NeuS [364], we incorporate an *Eikonal regularization* to ensure the neural field is a valid signed distance field: $L_{\text{eikonal}} = \sum_{\mathbf{x}} (\|\nabla_{\mathbf{x}} S\|_2 - 1)^2$,

Visibility optimization: Ambient occlusion and self-cast shadows pose challenges to the accuracy of inverse rendering, as it's difficult to separate them from albedo when optimizing photometric loss. However, with an estimated geometry, we can already obtain a strong visibility cue. Consequently, we utilize ambient occlusion mapping to prebake the visibility map onto the object surfaces obtained from the previous stage. We then minimize the visibility consistency term to ensure the rendered visibility $v_{\mathbf{r}}$ from MLP V aligns with the derived visibility $\hat{v}_r$: $\min_V \sum_{\mathbf{r}} \text{BCE}(v_{\mathbf{r}}, \hat{v}_{\mathbf{r}})$.

Material and illumination optimization: In the final stage, given the obtained surface geometry and the visibility network, we jointly optimize the environmental

light and the PBR material network. The overall objective is as follows:

$$\min_{M,\boldsymbol{\omega},\boldsymbol{\phi}} E_{\text{color}} + E_{\text{latent}} + E_{\text{metal}}, \tag{6.5}$$

where the three terms are defined as follows:

- The *color consistency term* E_{color} minimizes the discrepancy between the rendered color and observed color, akin to the geometry optimization stage. However, we use PBR-shaded color in place of the color queried from the auxiliary radiance field.

- The *latent code regularization* E_{latent} constrains the latent code $\boldsymbol{\rho}$ of the material network to closely align with a constant target vector $\boldsymbol{\rho}'$: $E_{\text{latent}} = \text{KL}\left(\boldsymbol{\rho}||\boldsymbol{\rho}'\right)$. We set $\boldsymbol{\rho}' = 0.05$.

- Lastly, inspired by the fact that most common objects are either metallic or not, we introduce a *metallic regularization* L_{metal} to encourage the predicted metallic value to be close to either 0 or 1: $L_{\text{metal}} = \sum_{\mathbf{r}} m_{\mathbf{r}}(1 - m_{\mathbf{r}})$.

6.4 Experiment

In this section, we evaluate the effectiveness of our model on synthetic and real-world datasets, analyze its characteristics, and showcase its applications.

6.4.1 Experiment setups

Data: Since existing multi-view datasets do not contain duplicate objects, we collect ***Dup***, a novel inverse rendering dataset featuring various duplicate objects. *Dup* consists of 13 synthetic and 6 real-world scenes, each comprising 5-10 duplicate objects such as cans, bottles, fire hydrants, etc. For synthetic data, we acquire 3D assets from PolyHaven[†] and utilize Blender Cycles for physics-based rendering. We generate 10-300 images per scene. As for the real-world data, we place the objects in different

[†]https://polyhaven.com/models

environments and capture 10-15 images using a mobile phone. The data allows for a comprehensive evaluation of the benefits of including duplicate objects in the scene for inverse rendering.

Metrics: Following prior art [430, 424], we employ Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), and LPIPS [426] to assess the quality of rendered and relit images. For materials, we utilize PSNR to evaluate albedo, and mean-squared error (MSE) to quantify roughness and environmental lighting. And following[364, 408, 249], we leverage the Chamfer Distance to measure the distance between our estimated geometry and the ground truth.

Baselines: We compare against three state-of-the-art multi-view inverse rendering approaches: Physg [424], InvRender [430], and NVdiffrec [249]. Physg and InvRender employ implicit representations to describe geometric and material properties, while NVdiffrec utilizes a differentiable mesh representation along with UV textures. Additionally, we enhance Physg by equipping it with a spatially-varying roughness.

Implementation details: We use the Adam optimizer with an initial learning rate 2e-4. All experiments are conducted on a single A100. The first stage takes 20 hours, and the 2nd and 3rd stage takes about 2 hours.

6.4.2 Experimental results

Single-view inverse rendering: We first evaluate our approach on the single-image multi-object setup. Since the baselines are not designed for this particular setup, we randomly select another 9 views, resulting in a total of 10 multi-view images, to train them. As shown in Tab. 6.2, we are able to leverage duplicate objects to constrain the underlying geometry, achieving comparable performance to the multi-view baselines. The variations in lighting conditions across instances also aid us in better disentangling the effects of lighting from the materials.

	Albedo PSNR ↑	Roughness MSE ↓	Relighting PSNR ↑	Env Light MSE ↓	Geometry CD ↓
M-S	20.229	0.096	21.328	**0.045**	0.010
S-M	**23.448**	**0.050**	**24.254**	0.052	**0.007**

Table 6.5: **Multi-view single object (M-S) vs single-view multi-object (S-M).**

	Albedo PSNR ↑	Roughness MSE ↓	Relighting PSNR ↑	Env. Light MSE ↓	Geometry CD ↓
Oracle	**17.858**	0.105	21.132	0.063	**0.031**
Ours	17.629	**0.062**	**21.374**	**0.051**	0.034

Table 6.6: **Single-view inverse rendering with estimated/ground-truth object poses.** Our model achieves similar results to the oracle model, suggesting that the pose estimation error is negligible.

Multi-view inverse rendering: Our method can be naturally extended to the multi-view setup, allowing us to validate its effectiveness in traditional inverse rendering scenarios. We utilize synthetic data to verify its performance. For each scene, we train our model and the baselines using 100 different views and evaluate the quality of the reconstruction results. Similar to previous work, we assume the ground truth poses are provided. As shown in Tab. 6.1, we outperform the baselines on all aspects. We conjecture that this improvement stems from our approach explicitly incorporating a material- and geometry-sharing mechanism during the modeling process. As a result, we have access to a significantly larger number of "effective views" during training compared to the baselines. We show some qualitative results in Fig. 6-4(left).

Real-world single-view inverse rendering: Up to this point, we have showcased the effectiveness of our approach in various setups using synthetic data. Next, we evaluate our model on real-world data. Due to the challenge of obtaining highly accurate ground truth for materials and geometry, we focus our comparison solely on the rendering results. As indicated in Tab. 6.4, our method achieves comparable performance to the multi-view baselines, even when trained using only a single view. We visualize some results in Fig. 6-1 and Fig. 6-6.

6.4.3 Analysis

Importance of the number of duplicate objects: Since our model utilize duplicate objects as a prior for 3D reasoning, one natural question to ask is: how many duplicate objects do we need? To investigate this, we randomly select a synthetic scene with 10 duplicates and gradually reduce the number of objects. We train our model under each setup and report the performance in Tab. 6.3. As expected, increasing the number of duplicates improves the accuracy of both material and geometry, since it provides more constraints on the shared object intrinsics.

Multi-view single object (M-S) vs. single-view multi-objects (S-M): Is observing an object from multiple views equivalent to observing multiple objects from a single view? Which scenario provides more informative data? To address this question, we first construct a scene containing 10 duplicate objects. Then, we place the same object into the same scene and capture 10 multi-view images. We train our model under both setups. Remarkably, the single-view setting outperforms the multi-view setting in all aspects (see Tab. 6.5). We conjecture this discrepancy arises from the fact that different instances of the object experience environmental lighting from various angles. Consequently, we are better able to disentangle the lighting effects from the material properties in the single-view setup. Fig. 6-5(right) shows some qualitative results.

Importance of 6 DoF pose estimation: Since our method adopts a stage-wise inference procedure, errors in pose estimation can propagate and impact the quality of the inverse rendering reconstructions. To verify the extent of this impact, we conduct an oracle experiment where we replace the estimated 6 DoF object poses with ground truth. As shown in Tab. 6.6, the average performance of the oracle model is similar to ours, suggesting that our pose estimation is very accurate. In fact, our pose estimation method achieves an average rotation error of 0.729° and a translation error of 1.522°.

Applications: Our approach supports various scene edits. Once we recover the material and geometry of the objects, as well as the illumination of the scene, we can faithfully relight existing objects, edit their materials, and seamlessly insert new objects into the environment as if they were originally present during the image capturing process (see Fig. 6-1).

Limitations: One major limitation of our approach is that we require the instances in each image to be nearly identical. Our method struggles when there are substantial deformations between different objects, as we adopt a geometry/material-sharing strategy. One potential way to address this is to loosen the sharing constraints and model instance-wise variations. However, increasing the capacity of the model may result in overfitting. We leave this for future study. Additionally, our approach currently requires decent 6 DoF poses as input and keeps the poses fixed. We could potentially combine it with BARF [211] to further refine our estimations.

6.5 Conclusion

We introduce a novel inverse rendering approach for single images with duplicate objects. We exploit the repetitive structure to estimate the 6 DoF poses of the objects, and incorporate a geometry and material-sharing mechanism to enhance the performance of inverse rendering. Experiments show that our method outperforms baselines, achieving highly detailed and precise reconstructions.

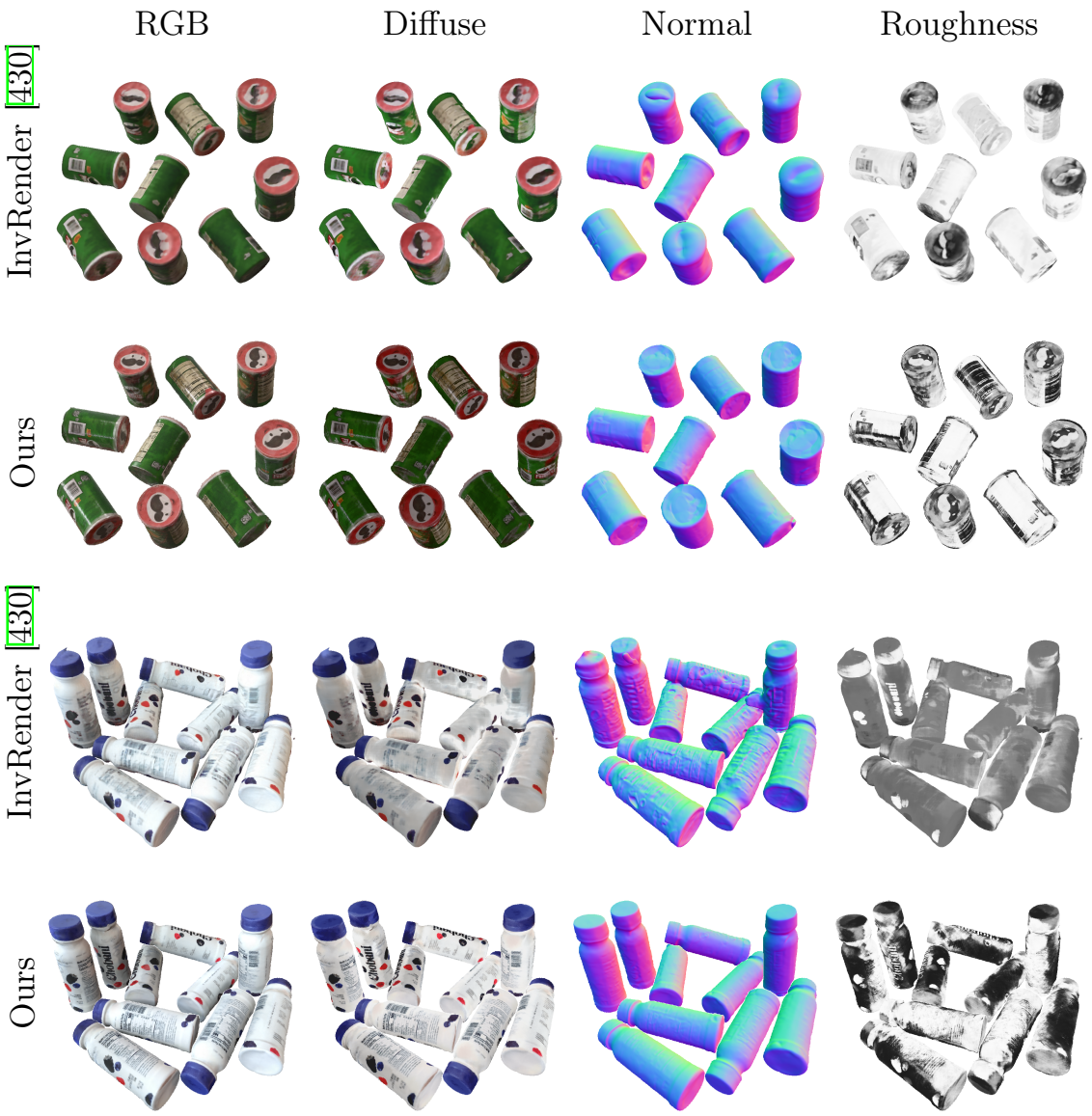


Figure 6-6: **Qualitative comparison on real-world data.** InvRender [430] takes as input 10 images, while we only consider one single view. Yet our approach is able to recover the underlying geometry and materials more effectively.

Chapter 7

3D Simulation and Generation

SGAM: Building a Virtual 3D World through Simultaneous Generation and Mapping

Yuan Shen, Wei-Chiu Ma, Shenlong Wang;

NeurIPS 2023.

UniSim: A Neural Closed-Loop Sensor Simulator

Ze Yang*, Yun Chen*, Jingkang Wang*, Sivabalan Manivasagam*, Wei-Chiu Ma, Anqi Joyce Yang, Raquel Urtasun;

CVPR 2023.

UltraLiDAR: Learning Compact Representations for LiDAR Completion and Generation

Yuwen Xiong, Wei-Chiu Ma, Jingkang Wang, Raquel Urtasun;

CVPR 2023.

We have now equipped machines with the ability to reason about various 3D properties under different conditions. Nevertheless, to truly comprehend the 3D world, intelligent systems must also be capable of hallucinating novel scenarios for better decision-making. In this chapter, we introduce two distinct methodologies to construct intelligent systems that can not only faithfully replicate the world, but also effectively simulate

counterfactual "what-if" scenarios. We show that with careful design one is able to, either explicitly or implicitly, build an editable digital twin of the world and enable various applications.

7.1 Introduction

Human perception goes way beyond simple recognition and reconstruction. Our extraordinary abilities not only allow us to make sense of what we see, but also enable us to imagine what we do not (*e.g.*, reason about what the scene looks like under different conditions). With a simple glance, we can effortlessly re-build a mental world, that may not be exactly like the original, but is perceived by our brain to be the same. In fact, it is such a profound ability drawing us apart from existing AI systems.

In this chapter, we aim to equip computational machines with similar capabilities. Our goal is to develop an intelligent 3D system that can not only faithfully reproduce what it "sees," but also realistically simulate temporally and spatially consistent sensory data (e.g., images, LiDAR point clouds) at novel viewpoints and for different scene configurations (e.g., actors at new placements). The task is of paramount interest to many applications in computer vision, computer graphics, geography, and robotics, since it unlocks numerous potentials. For instance, it may allow us to build an interactive virtual environment without any costly and laborious 3D modeling pipeline. We can not only synthesize novel, high-fidelity observations of the world for content creation, but also generate potentially infinite synthetic yet realistic training data for machine learning models. Importantly, it will allow us to test the policy of autonomous systems on a variety of scenarios, including hazardous long-tail situations that are difficult to test safely, without needing to deploy to the real world.

Indeed, there has been a consistent pursuit within the community in the past few decades [85, 84, 439, 325, 52, 341], where people attempt to design algorithms and models that are capable of extrapolating and hallucinating the world from existing observations. Unfortunately, a large body of efforts have been focusing on modeling in the 2D image space [85, 84, 439, 93, 212, 52], rely on densely collected observations

and focus on static scenes [246, 299, 300, 20, 214], or require external 3D assets and manually specified rules for restricted 3D simulation [193, 59, 9]. Recently, with the advent of deep generative models [175, 385, 216, 303, 301], researchers have made great strides in unconstrained 3D syntheses. However, due to the highly complex structure of the task, these approaches primarily focus on object-centric scenarios or the generation of small-scale environments. The ability to re-simulate a large-scale, dynamic 3D world and generate novel "what-if" scenarios remains elusive.

With these challenges in mind, we present two separate efforts towards 3D simulation and generation. We focus on the challenging self-driving scenario where the observations are sparse and often captured from constrained viewpoints (*e.g.*, straight trajectories along the roads). The first endeavor is to *explicitly* build an editable digital twin of the real world (through the data we captured), where existing actors in the scene can be modified or removed, new actors can be added, and new autonomy trajectories can be executed. This will enable the autonomy system to interact with the simulated world, where it receives new sensor observations based on its new location and the updated states of the dynamic actors, in a closed-loop fashion. Through a series of enhancements over prior neural rendering approaches, we are able to reconstruct and render multi-sensor (*i.e.*, LiDAR and camera) data for novel views and new scene configurations. Fig. 7-1 showcases the capabilities of our digital twin.

Our second research thrust is to learn the scene statistics from data *implicitly* and build an interpretable and controllable 3D generative model capable of generating entirely new scenes and structure from scratch. We focus mainly on LiDAR point clouds since it is the preferred data modality of most autonomous systems. Our key idea is to learn a *compact, discrete* 3D representation (codebook) of LiDAR point clouds that encodes the geometric structure of the scene and the physical rules of our world (*e.g.*, occlusion). Then, by learning a prior over the discrete codebook, we can generate novel, realistic driving scenes by sampling from it; we can also manipulate the discrete code of the scene and produce counterfactual scenarios, both of which can drastically improve the diversity and amount of LiDAR data. Fig. 7-2 shows some example outputs of our generative model.

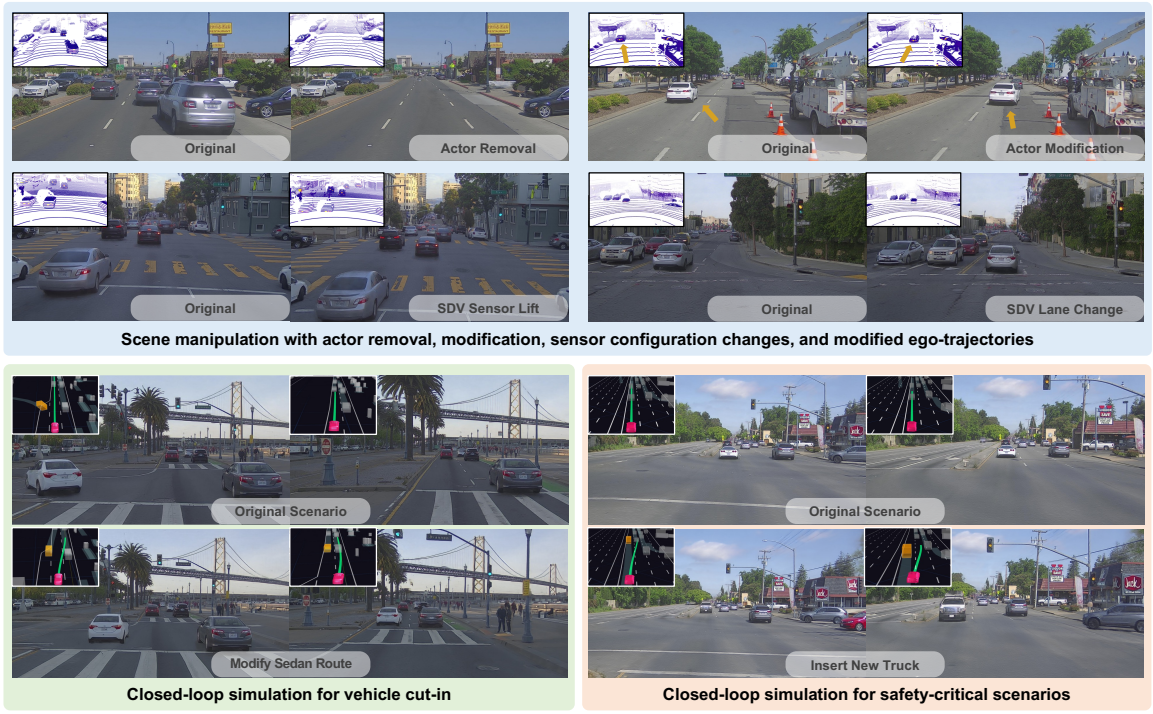


Figure 7-1: **Capabilities of our digital twin. Top:** We take recorded sensor data from a data collection platform and creates manipulable digital twins. **Bottom:** Our digital twin generates realistic, temporally consistent sensor simulations for new scenarios, enabling closed-loop autonomy evaluation. The autonomy system reactively interacts with the scenario, receives new sensor data, and changes lanes (see planned trajectory inset). All images and LiDAR are simulated .

We demonstrate the effectiveness of our proposed methodologies on two separate tasks: 3D simulation and 3D generation. For 3D simulation, our approach can realistically simulate camera and LiDAR observations at new views for large-scale dynamic driving scenes, achieving state-of-the-art performance in photorealism. Moreover, we reduce the domain gap over existing camera simulation methods on the downstream autonomy tasks of detection, motion forecasting and motion planning. Our simulated results can also be used to augment training data to improve perception models and perform closed-loop evaluation of autonomous systems on safety-critical scenarios. As for 3D generation, we compare our results with state- of-the-art LiDAR generative models. Our generated point clouds better match the statistics of those of ground truth data. We also conducted a human study where participants prefer our method over prior art over 98.5% (best 100%) of the time; comparing to ground truth, our

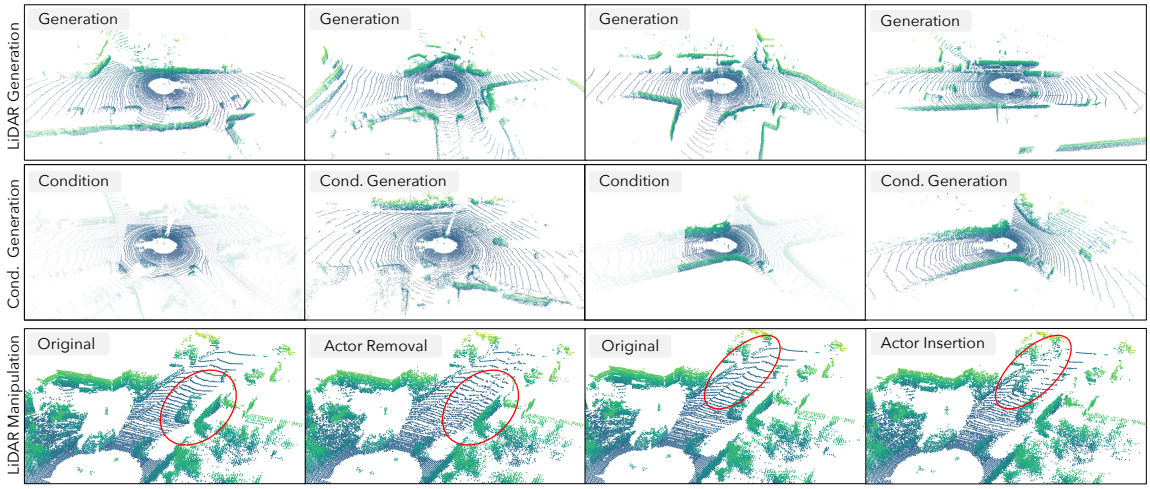


Figure 7-2: **Capabilities of our 3D generative model. Top row:** Diverse LiDAR generation with realistic global structure and fine-grained details; **Middle row:** Conditional scene generation with partially observed point clouds (highlighted in red). **Bottom row:** Controllable manipulation of real LiDAR with actor removal and insertion. Please see supp. material for more examples.

results were selected 32% of the time (best 50%).

7.2 Related Work

7.2.1 Simulation

Novel view synthesis: Recent novel view synthesis (NVS) work has achieved success in automatically generating highly photorealistic sensor observations [299, 246, 265, 181, 293, 266, 7, 223]. Such methods aim to learn a scene representation from a set of densely collected observed images and render the scene from nearby unseen viewpoints. Some works perform geometry reconstruction and then warp and aggregate pixel-features from the input images into new camera views, which are then processed by learning-based modules [299, 300, 7, 284]. Others represent the scene implicitly as a neural radiance field (NeRF) and perform volume rendering with a neural network [246, 20, 356, 403]. These methods can represent complex geometry and appearance and have achieved photorealistic rendering, but focus on small static

scenes. Several representations [218, 48, 239, 254, 291, 292, 340, 248, 423] partition the space and model the volume more efficiently to handle large-scale unbounded outdoor scenes. However, these works focus primarily on the NVS task where a dense collection of images are available and most test viewpoints are close to the training views, and focus on the static scene without rendering dynamic objects such as moving vehicles. In contrast, our work extends NVS techniques to build a sensor simulator from a single recorded log captured by a high-speed mobile platform. We aim to render image and LiDAR observations of dynamic traffic scenarios from new viewpoints and modified scene configurations to enable closed-loop autonomy evaluation.

Data-driven sensor simulation for self-driving: Several past works have leveraged computer vision techniques and real world data to build sensor simulators for self-driving. Some works perform 3D reconstruction by aggregating LiDAR and building textured geometry primitives for physics-based rendering [338, 237, 95, 405], but primarily simulate LiDAR or cannot model high-resolution images. Another line of work perform object reconstruction and insertion into existing images [59, 378, 403, 362] or point clouds [361, 94, 401, 400], but these methods are unable to render sensor data from new views for closed-loop interaction. DriveGAN [171] represents the scene as disentangled latent codes and generates video from control inputs with a neural network for differentiable closed-loop simulation, but is limited in its realism and is not temporally consistent. AADS [193] and VISTA 2.0 [9, 8, 373], perform multi-sensor simulation via image-based warping or ray-casting on previously collected sensor data to render new views of the static scene, and then insert and blend CAD assets into the sensor data to create new scenarios. These approaches, while promising, have visual artifacts for the inserted actors and rendered novel views, resulting in a large domain gap. Neural Scene Graphs (NSG) [265] and Panoptic Neural Fields (PNF) [181] represent the static scene and agents as multi-layer perceptrons (MLPs) and volume render photorealistic images of the scene. However, the single MLP has difficulties modelling large scale scenes. These prior works also focus on scene editing and perception tasks where the SDV does not deviate significantly from the original

recording. Instead, we focus on multi-sensor simulation for closed loop evaluation of autonomy systems, and specifically design our system to better handle extrapolation.

7.2.2 Generation

Image generation: How to synthesize an image realistically has been a long-standing problem in computer vision. The task dates back to 60s [159, 29] where researchers attempted to generate textures by matching statistics [235, 130]. Through parametric sampling [439] or non-parametric matching [85, 84], they were able to synthesize an infinite amount of high-fidelity texture images. Unfortunately, these approaches fall short when applied to natural images, since the images have much higher complexity. Recently, with the help of deep generative models [108, 174, 283, 134, 328], researchers have demonstrated promising results on generating photo-realistic images [148, 165, 166]. With proper design and inductive biases [354, 93, 62], they are even able to scale the output to mega-pixel level [344, 52]. In this chapter, we take inspiration from latest 2D generative models [93, 52] and build on top of it. However, instead of treating 3D generation as a 2D task (*e.g.* range image, depth map) as in the past, we explicitly model the 3D geometric relationship — both at the input level and the output level. By encoding 3D information into the quantized codebook, we are able to synthesize a large amount of high-quality, realistic point clouds for large-scale outdoor environment, which are not exactly like the original, but will be perceived by humans to be real.

3D Generation: 3D modeling and synthesis have been an active yet challenging research problem for decades [46, 32, 341]. Recently, with the development of image generation techniques [108, 174, 297], the field has experienced a rapid growth [391, 258, 343]. Drawing inspiration from its corresponding 2D analogue, researchers have been able to generate high quality point clouds [206, 398, 40], voxels [104, 392], meshes [113, 106], etc. Unfortunately, since 3D solution space is much more intricate than that of 2D, these approaches are typically object-centric; also, they mostly focus on generating common objects in our daily lives [114, 209, 81, 82]. To enable scene-level

synthesis, researchers have sought to incorporate more structures into the generation pipeline [253, 281, 270, 280], such as reducing the output space from full 3D to predefined, compact representation [274, 252, 339]. While these strategies greatly alleviate the issue, the generated scene scale is still rather limited (*e.g.*, an indoor environment). In this chapter, we push the boundary of 3D generation systems and present a approach that is capable of generating large-scale, coherent 3D scene structure more than a hundred thousand square feet. The produced high-quality 3D world is interpretable, easy to manipulate, and captures the physical structure of the real world (*e.g.* occlusion patterns).

7.3 Method I: Building Editable Digital Twins

7.3.1 Overview

Given a log with camera images and LiDAR point clouds captured by a moving platform, as well as their relative poses in a reference frame, our goal is to construct an *editable* and *controllable* digital twin, from which we can generate realistic multi-modal sensor simulation and counterfactual scenarios of interest. We build our model based on the intuition that the 3D world can be decomposed as a static background and a set of moving actors. By effectively disentangling and modeling each component, we can manipulate the actors to generate new scenarios and simulate the sensor observations from new viewpoints. Towards this goal, we propose a neural rendering closed-loop simulator that jointly learns shape and appearance representations for both the static scene and dynamic actors from the sensor data captured from a single pass of the environment.

We unfold this section by first reviewing the basic building blocks of our approach. Next, we present our compositional scene representation, and detail how we design our background and dynamic actor models. We then describe how to generate simulated sensor data with. Finally, we discuss how to learn the model from real-world data. Fig. 7-3 shows an overview of our approach.

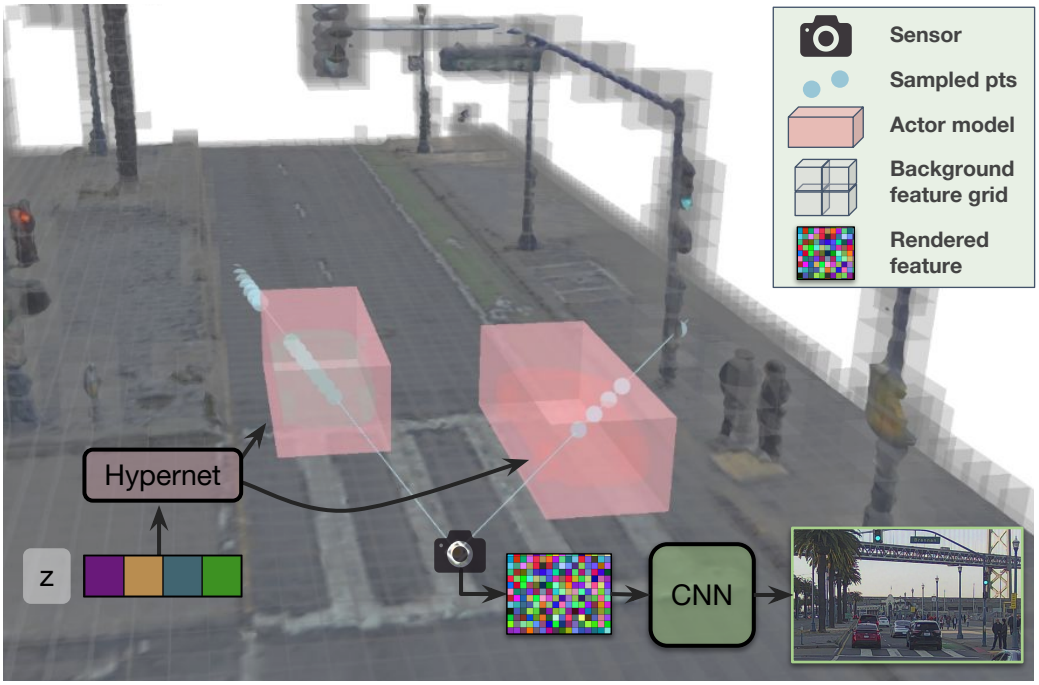


Figure 7-3: **Overview of our digital twin:** We divide the 3D scene into a static background (grey) and a set of dynamic actors (red). We query the neural feature fields separately for static background and dynamic actor models, and perform volume rendering to generate neural feature descriptors. We model the static scene with a sparse feature-grid and use a hypernetwork to generate the representation of each actor from a learnable latent. We finally use a convolutional network to decode feature patches into an image.

7.3.2 Preliminaries

Neural feature fields: A feature field refers to a continuous function f that maps a 3D point $\mathbf{x} \in \mathbb{R}^3$ and a view direction $\mathbf{d} \in \mathbb{R}^2$ to an implicit geometry $s \in \mathbb{R}$ and a N_f-dimensional feature descriptor $\mathbf{f} \in \mathbb{R}^{N_f}$. Since the function is often parameterized as a neural network $f_\theta : \mathbb{R}^3 \times \mathbb{R}^2 \rightarrow \mathbb{R} \times \mathbb{R}^{N_f}$, with θ the learnable weights, we call it neural feature field (NFF). NFFs can be seen as a superset of several existing works [246, 244]. If we represent the implicit geometry as volume density $s \in \mathbb{R}^+$ and the feature descriptor as RGB radiance $\mathbf{f} \in \mathbb{R}^3$, NFFs become NeRFs [246]. If we enforce the implicit geometry to be the probability of occupancy, NFFs become occupancy functions [244]. Importantly, NFFs naturally support composition [116, 259, 181], enabling the combination of multiple relatively simple NFFs to form a complex field.

Multi-resolution features grid: To improve the expressiveness and speed of NFFs, past works [337, 248, 63, 412] further combined learnable multi-resolution features grid $\{\mathcal{G}^l\}_{l=1}^L$ with a neural network f. Specifically, given a query point $\mathbf{x} \in \mathbb{R}^3$, the 3D feature grid at each level is first trilinearly interpolated. The interpolated features are then concatenated with the view direction $\mathbf{d} \in \mathbb{R}^2$, and the resulting features are processed with an MLP head to obtain the geometry s and feature descriptor $\mathbf{f}$:

$$s, \mathbf{f} = f\left(\{\texttt{interp}(\mathbf{x}, \mathcal{G}^l)\}_{l=1}^L, \mathbf{d}\right). \tag{7.1}$$

These multi-scale features encode both global context and fine-grained details, providing richer information comparing to the original input $\mathbf{x}$. This also enables using a smaller f, which significantly reduces the inference time [330, 337]. In practice, we optimize the features grid using a fixed number of features $\mathcal{F}$, and map the features grid $\{\mathcal{G}^l\}_{l=1}^L$ to $\mathcal{F}$ with a grid index hash function [248]. Hereafter, we will use $\mathcal{F}$ and $\{\mathcal{G}^l\}_{l=1}^L$ interchangeably.

7.3.3 Compositional Neural Scene Representation

We aim to build a compositional scene representation that best models the 3D world including the dynamic actors and static scene. Given a recorded log captured by a data collection platform, we first define a 3D space volume over the traversed region. The volume consists of a static background $\mathcal{B}$ and a set of dynamic actors $\{\mathcal{A}_i\}_{i=1}^N$. Each dynamic actor is parameterized as a bounding box of dimensions $\mathbf{s}_{\mathcal{A}_i} \in \mathbb{R}^3$, and its trajectory is defined by a sequence of SE(3) poses $\{\mathbf{T}_{\mathcal{A}_i}^t\}_{t=1}^T$. We then model the static background and dynamic actors with separate multi-resolution features grid and NFFs. Let the static background be expressed in the world frame. We represent each actor in its object-centroid coordinate system (defined at the centroid of its bounding box), and transform their features grid to world coordinates to compose with the background. This allows us to disentangle the 3D motion of each actor, and focus on representing shape and appearance. To learn high-quality geometry [364, 408], we adopt the signed distance function (SDF) as our implicit geometry representation s.

We now describe each component in more detail.

Sparse background scene model: We model the whole static scene with a multi-resolution features grid $\mathcal{F}_{\text{bg}}$ and an MLP head f_{bg}. Since a self-driving log often spans hundreds to thousands of meters, it is computationally and memory expensive to maintain a dense, high-resolution voxel grid. We thus utilize geometry priors from LiDAR observations to identify near-surface voxels and optimize only their features. Specifically, we first aggregate the static LiDAR point cloud from each frame to construct a dense 3D scene point cloud. We then voxelize the scene point cloud and obtain a scene occupancy grid $\mathbf{V}_{\text{occ}}$. Finally, we apply morphological dilation to the occupancy grid and coarsely split the 3D space into free *vs*. near-surface space. As the static background is often dominated by free space, this can significantly sparsify the features grid and reduce the computation cost. The geometric prior also allows us to better model the 3D structure of the scene, which is critical when simulating novel viewpoints with large extrapolation. To model distant regions, such as sky, we follow [423, 21] to extend our background scene model to unbounded scenes.

Generalized actor model: One straightforward way to model the actors is to parameterize each actor $\mathcal{A}_i$ with a features grid $\mathcal{F}_{\mathcal{A}_i}$ and adopt a shared MLP head $f_{\mathcal{A}}$ for all actors. In this design, the individual features grid encodes instance-specific geometry and appearance, while the shared network maps them to the same feature space for downstream applications. Unfortunately, such a design requires large memory for dense traffic scenes and, in practice, often leads to overfitting — the features grid does not generalize well to unseen viewpoints. To overcome such limitations, we propose to learn a hypernetwork [117] over the parameters of all grids of features. The intuition is that different actors are observed from different viewpoints, and thus their grids of features are informative in different regions. By learning a prior over them, we can capture the correlations between the features and infer the invisible parts from the visible ones. Specifically, we model each actor $\mathcal{A}_i$ with a low-dimensional latent

code $\mathbf{z}_{\mathcal{A}_i}$ and learn a hypernetwork $f_{\mathbf{z}}$ to regress the features grid $\mathcal{F}_{\mathcal{A}_i}$:

$$\mathcal{F}_{\mathcal{A}_i} = f_{\mathbf{z}}(\mathbf{z}_{\mathcal{A}_i}). \tag{7.2}$$

Similar to the background, we adopt a shared MLP head $f_{\mathcal{A}}$ to predict the geometry and feature descriptor at each sampled 3D point via Eq. 7.1. We jointly optimize the actor latent codes $\{\mathbf{z}_{\mathcal{A}_i}\}$ during training.

Composing neural feature fields: Inspired by works that composite solid objects [116, 265] into a scene, we first transform object-centric neural fields of the foreground actors to world coordinates with the desired poses (*e.g.*, using $\mathbf{T}^t_{\mathcal{A}_i}$ for reconstruction). As the static background is a sparse features grid, we then simply replace the free space with the actor feature fields. Through this simple operation, we can insert, remove, and manipulate the actors within the scene.

7.3.4 Multi-modal Sensor Simulation

Now that we have a composed scene representation of the static and dynamic world, the next step is to render it into the data modality of interest. In this work, we focus on camera images and LiDAR point clouds, as they are the two main sensory modalities employed by modern SDVs.

Camera simulation: Following recent success in NVS [259, 49], we adopt a hybrid volume and neural rendering framework for efficient photorealistic image simulation. Given a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ shooting from the camera center $\mathbf{o}$ through the pixel center in direction $\mathbf{d}$, we first sample a set of 3D points along the ray and extract their features and geometry (Eq. 7.1). We then aggregate the samples and obtain a pixel-wise feature descriptor via volume rendering:

$$\mathbf{f}(\mathbf{r}) = \sum_{i=1}^{N_{\mathbf{r}}} w_i \mathbf{f}_i, \quad w_i = \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{7.3}$$

Here, $\alpha_i \in [0, 1]$ represents opacity, which we can derive from the SDF s_i using an approximate step function $\alpha = 1/(1 + \exp(\beta \cdot s))$, and β is the hyper-parameter controlling the slope. We volume render all camera rays and generate a 2D feature map $\mathbf{F} \in \mathbb{R}^{H_f \times W_f \times N_f}$. We then leverage a 2D CNN g_{rgb} to *render* the feature map to an RGB image $\mathbf{I}_{\text{rgb}}$:

$$g_{\text{rgb}} : \mathbf{F} \in \mathbb{R}^{H_f \times W_f \times N_f} \rightarrow \mathbf{I}_{\text{rgb}} \in \mathbb{R}^{H \times W \times 3}. \tag{7.4}$$

In practice, we adopt a smaller spatial resolution for the feature map $H_f \times W_f$ than that of the rendered image $H \times W$, and rely on the CNN g_{rgb} for upsampling. This allows us to significantly reduce the amount of ray queries.

LiDAR simulation: LiDAR point clouds encode 3D (depth) and intensity (reflectivity) information, both of which can be simulated in a similar fashion to Eq. 7.3. We assume the LiDAR to be a time-of-flight pulse-based sensor, and model the pulses transmitted by the oriented LiDAR laser beams as a set of rays. We slightly abuse the notation and let $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ be a ray casted from the LiDAR sensor we want to simulate. Denote $\mathbf{o}$ as the center of the LiDAR and $\mathbf{d}$ as the normalized vector of the corresponding beam. We then simulate the depth measurement by computing the expected depth of the sampled 3D points:

$$D(\mathbf{r}) = \sum_{i=1}^{N_{\mathbf{r}}} w_i t_i. \tag{7.5}$$

As for LiDAR intensity, we volume render the ray feature (using Eq. 7.3) and pass it through an MLP intensity decoder g_{int} to predict its intensity $l^{\text{int}}(\mathbf{r}) = g_{\text{int}}(\mathbf{f}(\mathbf{r}))$.

7.3.5 Learning

We jointly optimize all grids of features $\mathcal{F}_*$ (including latent codes $\{\mathbf{z}_{\mathcal{A}_i}\}$, the hypernetwork $f_{\mathbf{z}}$, the MLP heads $(f_{\text{bg}}, f_{\mathcal{A}})$ and the decoders $(g_{\text{rgb}}, g_{\text{int}})$ by minimizing the difference between the sensor observations and our rendered outputs. We also

regularize the underlying geometry such that it satisfies real-world constraints. Our full objective is:

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \lambda_{\text{lidar}}\mathcal{L}_{\text{lidar}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}}.$$

In the following, we discuss in more detail each term.

Image simulation $\mathcal{L}_{\text{rgb}}$: This objective consists of a ℓ_2 photometric loss and a perceptual loss [427, 372], both measured between the observed images and our simulated results. We compute the loss in a patch-wise fashion:

$$\mathcal{L}_{\text{rgb}} = \frac{1}{N_{\text{rgb}}} \sum_{i=1}^{N_{\text{rgb}}} \left(\left\| \mathbf{I}_i^{\text{rgb}} - \hat{\mathbf{I}}_i^{\text{rgb}} \right\|_2 + \lambda \sum_{j=1}^{M} \left\| V^j(\mathbf{I}_i^{\text{rgb}}) - V^j(\hat{\mathbf{I}}_i^{\text{rgb}}) \right\|_1 \right), \tag{7.6}$$

where $\mathbf{I}_i^{\text{rgb}} = f_{\text{rgb}}(\mathbf{F}_i)$ is the rendered image patch (Eq. 7.4) and $\hat{\mathbf{I}}_i^{\text{rgb}}$ is the corresponding observed image patch. V^j denotes the j-th layer of a pre-trained VGG network [322].

LiDAR simulation $\mathcal{L}_{\text{lidar}}$: This objective measures the ℓ_2 error between the observed LiDAR point clouds and the simulated ones. Specifically, we compute the depth and intensity differences:

$$\mathcal{L}_{\text{lidar}} = \frac{1}{N} \sum_{i=1}^{N} \left(\left\| D(\mathbf{r}_i) - D_i^{\text{obs}} \right\|_2 + \left\| l^{\text{int}}(\mathbf{r}_i) - \hat{l}_i^{\text{int}} \right\|_2 \right). \tag{7.7}$$

Since LiDAR observations are noisy, we filter outliers and encourage the model to focus on credible supervision. In practice, we optimize 95% of the rays within each batch that have smallest depth error.

Regularization $\mathcal{L}_{\text{reg}}$: We further apply two additional constraints on the learned representations. First, we encourage the learned sample weight distribution w (Eq. 7.3) to concentrate around the surface. Second, we encourage the underlying SDF s to satisfy the Eikonal equation, which helps the network optimization find a smooth zero

Figure 7-4: **Qualitative comparison of image simulation.** We show simulated images in both the interpolation (rows 1, 3) and lane-shift test settings (rows 2, 4).

Methods	Interpolation			Lane Shift	
	PSNR↑	SSIM↑	LPIPS↓	FID↓ @ 2m	FID↓ @ 3m
FVS [299]	21.09	0.700	0.299	112.6	135.8
NSG [265]	20.74	0.600	0.556	319.2	343.0
Instant-NGP [248]	24.03	0.708	0.451	192.8	220.1
Ours	**25.63**	**0.745**	**0.288**	**74.7**	**97.5**

Table 7.1: **Quantitative comparison against prior art on image simulation.**

level set [111]:

$$\mathcal{L}_{\text{reg}} = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{\tau_{i,j}>\epsilon}\|w_{ij}\|_2 + \sum_{\tau_{i,j}<\epsilon}\left(\|\nabla s(\mathbf{x}_{ij})\|_2 - 1\right)^2\right), \tag{7.8}$$

where $\tau_{i,j} = |t_{ij} - D_i^{\text{gt}}|$ is the distance between the sample $\mathbf{x}_{ij}$ and its corresponding LiDAR observation D_i^{gt}.

Adversarial loss $\mathcal{L}_{\text{adv}}$: To improve photorealism at unobserved viewpoints, we train a discriminator CNN $\mathcal{D}_{\text{adv}}$ to differentiate between our simulated images at observed viewpoints and unobserved ones. Specifically, we denote the set of rays to render an image patch as $\mathbf{R} = \{\mathbf{r}(\mathbf{o}, \mathbf{d}_j)\}_{j=1}^{P\times P}$, and randomly jitter the ray origin to create unobserved ray patches $\mathbf{R}' = \{\mathbf{r}(\mathbf{o}+\epsilon, \mathbf{d}_j)\}_{j=1}^{P\times P}$, where $\epsilon \in \mathcal{N}(0, \sigma)$. The

Methods	Interpolation			Lane Shift	
	PSNR↑	SSIM↑	LPIPS↓	FID↓ @ 2m	FID↓ @ 3m
NFF only	24.93	0.717	0.393	153.7	173.5
+ Actor model	25.80	0.744	0.364	84.1	111.8
+ CNN	**25.99**	**0.762**	0.341	78.8	103.3
+ VGG & GAN loss	25.63	0.745	**0.288**	**74.7**	**97.5**

Table 7.2: **Contribution of each component on simulation.**

	Median ℓ_2 Error (m)↓	Hit Rate↑	Intensity RMSE↓
LiDARSim [237]	0.11	92.2%	0.091
Ours	**0.10**	**99.6%**	**0.065**

Table 7.3: **Quantitative comparison of LiDAR simulation.**

discriminator CNN $\mathcal{D}_{\text{adv}}$ minimizes:

$$-\frac{1}{N_{\text{adv}}}\sum_{i=1}^{N_{\text{adv}}}\log\mathcal{D}_{\text{adv}}(\mathbf{I}_i^{\text{rgb},\mathbf{R}})+\log(1-\mathcal{D}_{\text{adv}}(\mathbf{I}_i^{\text{rgb},\mathbf{R}'})), \tag{7.9}$$

where $\mathbf{I}_i^{\text{rgb},\mathbf{R}}=f_{\text{rgb}}(\mathbf{F}(\mathbf{R}_i))$ and $\mathbf{I}_i^{\text{rgb},\mathbf{R}'}=f_{\text{rgb}}(\mathbf{F}(\mathbf{R}_i'))$ are the rendered image patches at observed and unobserved viewpoints, respectively. We then define the adversarial loss $\mathcal{L}_{\text{adv}}$ to train the CNN RGB decoder g_{rgb} and neural feature fields to improve photorealism at unobserved viewpoints as:

$$\mathcal{L}_{\text{adv}}=\frac{1}{N_{\text{adv}}}\sum_{i=1}^{N_{\text{adv}}}\log(1-\mathcal{D}_{\text{adv}}(\mathbf{I}_i^{\text{rgb},\mathbf{R}'})). \tag{7.10}$$

Implementation details: We identify actors along rendered rays using the AABB ray-box intersection [234]. When sampling points along the ray, we adopt a larger step size for background regions and a smaller one for intersected actor models to ensure appropriate resolution. We leverage the scene occupancy grid $\mathbf{V}_{\text{occ}}$ to skip point samples in free space. During learning, we also optimize the actor trajectories to account for noise in the initial input. For vehicle actors, we also leverage the shape prior that they are symmetric along their length.

7.4 Method II: Learning Controllable Generative Models

In the previous section, we have shown how to explicitly leverage the compositional structure of the world to build an editable digital twin and manipulate it to simulate different scenarios. One natural follow-up question is: Can we directly learn all these structures from data and generate entirely new scenes and content from scratch? Such a 3D generative model will be beneficial for content creation and will also provide an endless playground for the robot to explore and learn.

With this motivation in mind, in this section, we seek to learn a compact 3D representation of scene-level LiDAR point clouds for realistic (un)conditional LiDAR generation and manipulation. Based on the observation that vector quantized (VQ) representations [354, 290, 92] are robust to noise, easy to manipulate, and naturally compatible with generative models, we propose to learn a discrete codebook for LiDAR point clouds and build our model on top of it.

We start by reviewing the basics of vector-quantized variational autoencoder (VQ-VAE) [354], a building block of our approach. Then we showcase how to exploit similar concepts to encode 3D point clouds into a discrete codebook. Finally, we discuss how to exploit the discrete representation for different tasks.

7.4.1 Discrete Representations for LiDAR

VQ-VAE revisit: The goal of VQ-VAE is to learn a discrete latent representation that is expressive, robust to noise, and compatible with generative models. VQ-VAE consists of three parts: (i) an encoder E that encodes the input signal, which for simplicity we assume to be an image, $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$ to a continuous embedding map $\mathbf{z} = E(\mathbf{x}) \in \mathbb{R}^{h \times w \times D}$, (ii) an element-wise quantization function q that maps each embedding to its closest learnable latent code $\mathbf{e}_k \in \mathbb{R}^D$, with $k = 1, ..., K$, and (iii) a decoder G that takes as input the quantized representation $\hat{\mathbf{z}} = q(\mathbf{z})$ and outputs the reconstructed image $\hat{\mathbf{x}} = G(\hat{\mathbf{z}})$. The whole model can be trained end-to-end by

minimizing:

$$\mathcal{L}_{\text{vq}} = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \|\text{sg}[E(\mathbf{x})] - \hat{\mathbf{z}}\|_2^2 + \|\text{sg}[\hat{\mathbf{z}}] - E(\mathbf{x})\|_2^2, \tag{7.11}$$

where sg$[\cdot]$ denotes the stop-gradient operation.

The limited number of discrete codes $\mathbf{e}$ stabilizes the input distribution of the decoder during training; it also forces the codes to capture meaningful, re-usable information as the decoder can no longer "seek shortcut" from the continuous signals for the reconstruction task. VQ-VAE has enjoyed great success across different modalities, such as natural images [92, 52], audio [354] and 2.5D images [319]. In this work, we further extend it to learn discrete representations for 3D LiDAR point clouds.

VQ-VAE for LiDAR: We aim to learn a discrete codebook that can effectively represent a set of LiDAR point clouds. Directly applying VQ-VAE, however, is challenging, since the fixed set of discrete latents will have to model point clouds that live in a continuous 3D space, and deal with the fact that each point cloud may have a different number of points. To address these issues, we propose to voxelize the point clouds and instead infer whether each voxel is occupied or not. By grounding the point clouds with a pre-defined grid (similar to the 2D pixel grid of images), we can foster the discrete codebook to learn the overall structure rather than the minor 3D positional variations. This representation also can naturally handle the varying number of points. While we may sacrifice some precision during the voxelization process, the impact is negligible for both LiDAR generation (see Sec. 7.5.2).

Now that we have a voxelized point cloud, the next step is to design the encoder E and the decoder G. For large scenes with high resolution, 3D convolution becomes very expensive since we need to infer the occupancy of each voxel densely. We thus convert the input to Birds-Eye-View (BEV) images by treating the height dimension of the voxel grids as feature channel C and then adopt 2D convolutions instead. In this case, we can process 3D LiDAR data just like 2D images; we can also exploit existing model architectures designed for 2D images directly. We note that such BEV

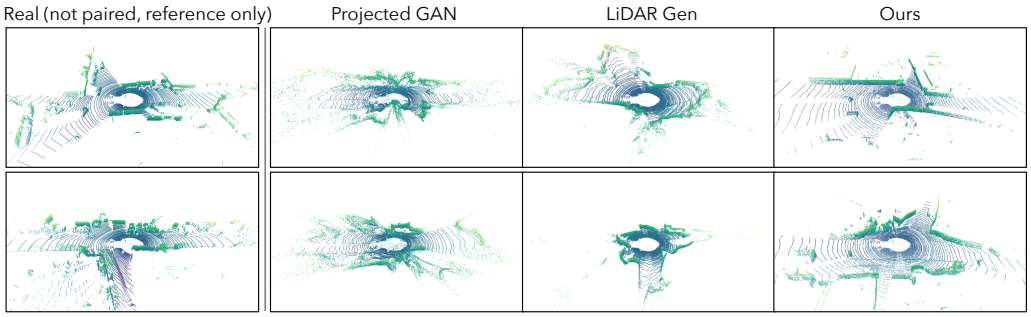


Figure 7-5: **Qualitative comparison against baselines on unconditional LiDAR generation.** We compare with two state-of-the-art LiDAR generation methods Projected GAN [312] and LiDARGen [442] and include real data for reference. Our model can generate results with more structured layouts and clearer beam patterns.

images have been widely adopted in the context of self-driving perception [58, 419], since they encode rich geometric information. The output of the decoder is a logit grid $\hat{\mathbf{x}} \in \mathbb{R}^{H \times W \times C}$. It can be further converted to a binary voxel grid $\hat{\mathbf{x}}^{\text{bin}} \in \{0, 1\}^{H \times W \times C}$ through gumbel softmax [150].

Finally, we train our LiDAR VQ-VAE model with Eq. 7.11, except that we replace the ℓ_2 reconstruction loss with a binary (occupied or not) cross-entropy loss. To improve the realism of the reconstruction, we further adopt a pre-trained voxel-based detector V and measure the feature difference, similar to perpectual loss [154]. Our full loss is:

$$\mathcal{L}_{\text{feat}} = \mathcal{L}_{\text{vq}} + \|V_{\text{b}}(\mathbf{x}) - V_{\text{b}}(\hat{\mathbf{x}}^{\text{bin}})\|_2^2. \tag{7.12}$$

V_{b} denotes the feature from the last backbone layer of V.

LiDAR manipulation: Once we train the model, we can easily manipulate arbitrary LiDAR point clouds by editing their corresponding latent codes. Since objects are spatially apart in 3D, the model can dedicate specific codes for them. We can thus identify the codes for objects of interest (*e.g.*, vehicles) and insert/remove them into/from the scene. As shown in Fig. 7-2, we can populate vehicles on the street and create counterfactual scenarios.

7.4.2 LiDAR Generation

As we have alluded to earlier, the learned discrete representations can be naturally combined with generative models and generate high-fidelity LiDAR point clouds, both unconditionally and conditionally.

Unconditional generation: Given the learned codebook $\mathbf{e}$ and the decoder G, the problem of LiDAR generation can be formulated as code map generation. Instead of directly generating LiDAR point clouds, we first generate discrete code maps in the form of code indices. Then we map the indices to discrete features by querying the codebook and decoding them back to LiDAR point clouds with the decoder. Following Chang *et al*. [52], we adopt a bi-directional self-attention Transformer [129, 52] to *iteratively* predict the code map. Specifically, we start from a blank canvas. At each iteration, we select a subset of the predicted codes with top confidence scores and update the canvas accordingly. With the help of the Transformer, we can aggregate context from the whole map and predict missing parts based on already predicted codes. In the end, the canvas will be filled with predicted code indices, from which we can decode LiDAR point clouds. We refer the reader to [52] for more details.

Conditional generation: Our unconditional LiDAR generation pipeline can be easily extended to perform conditional generation. Instead of starting the generation process from an empty canvas, one can simply start with a partially filled code map and let the Transformer predict the rest. For instance, we can place `[CAR]` codes at regions of interest; and run the model multiple times. We can then obtain different traffic scenarios with the pre-defined cars untouched.

Free space suppression sampling: Our iterative generation procedure can be viewed as a variant of coarse-to-fine generation. The codes generated during early iterations determine the overall structure, while the ones generated at the end are in charge of fine-grained details. While this pipeline is effective for image generation [52], it may lead to degenerated results when generating LiDAR point clouds. One

critical reason is that LiDAR point clouds are sparse, and a large portion of the scene is represented by the same [BLANK] codes. Since the [BLANK] codes occur frequently, the Transformer tends to predict them with high scores. If we naively sample the codes based on the output of the Transformer, we may fill most of the canvas with [BLANK] codes, and little structure will remain. To address this issue, we suppress the [BLANK] codes during the early generation stages by setting their probability to 0. This ensures the model generates meaningful structures in the beginning. We identify the [BLANK] codes by looking at the occurrence statistic of all codes across the whole dataset. We empirically select the top as [BLANK] codes.

Iterative denoising: With free space suppression sampling, we can already obtain good results. However, the generated point clouds sometimes still contain high-frequency noise (*e.g.*, there might be some floating points in the very far range). To mitigate this issue, we randomly mask out different regions of the output LiDAR point clouds and re-generate them. The intuition is that if we mask out a structured region, we can still recover it through the neighborhood context. However, if the masked region corresponds to pure noise that is irrelevant to the surrounding, it will likely be removed after multiple trials (since the model cannot infer it from the context).

Training: We first encode all LiDAR point clouds into frozen discrete representations (code maps) learned in Sec. 7.4.1. Then, at each training iteration, we randomly mask out a subset of codes. Finally, we adopt the bi-directional Transformer to predict the correct code for those masked regions. Since we have GT code map, we supervise the model with cross-entropy loss. See [52] for more details.

7.4.3 Implementation Details

In this section, we discuss implementation details that are crucial for learning discrete LiDAR representations.

Voxel sizes matter: We set the input voxel size to be $15.625 \times 15.625 \times 15$ cm for x, y, z dimensions. We find that the downsampling ratio when mapping the BEV image to the discrete code has a huge impact on both reconstruction and generation performance. If the patch size that each discrete code represents is too large, a single fixed code does not have enough capacity to model the variations (e.g., a small position/rotation shift of a car inside the patch). We empirically find that downsampling $8\times$ achieves a good trade-off between preserving high-frequency information in the LiDAR data and maintaining high-level semantic meaning. Thus the patch size is 8×8, leading to 1.25×1.25 m on the spatial dimension that each code represents.

Model hyperparameters: As a typical BEV image usually has a large spatial range, the resolution of the image and the number of codes are high. We use Swin Transformer [222] instead of the vanilla Vision Transformer [80] to reduce the computational cost for our generative Transformer model. It has 24 layers and 8 heads, and the embedding dimension is set to 512. All other training hyper-parameters like optimizer settings and label smoothing are kept the same as in [52]. For simplicity, we use the same architecture in our VQ-VAE learning. The encoder and decoder are both Swin Transformers with 12 layers, respectively. We set the codebook size to 1024 with 1024 hidden dimensions for each code.

Codebook (re)-initialization: We empirically find that the codebook can easily collapse (only a few codes are used) during training. For better codebook learning, we use data-dependent codebook initialization. Specifically, we use a memory bank to store the continuous embedding output from the encoders at each iteration; and use K-Means centroids of the memory bank to initialize/reinitialize the codebook if the code utilization percentage is lower than a threshold (empirically, we define a code is not used for 256 iterations as "dead code" and set the threshold to be 50%). During the first 2000 iterations of training, we gradually shifted the decoder input from continuous to quantized embeddings as a warmup. We find these strategies help achieve good codebook learning and utilization.

7.5 Experiments

7.5.1 3D Simulation

Dataset: We evaluate our method on the publicly available PandaSet [393] which contains 103 driving scenes captured in urban areas in San Francisco. Each scene is composed of 8 seconds (80 frames, sampled at 10hz) of images captured from a front-facing wide angle camera (1920×1080) and point clouds from 360° spinning LiDAR.

Baselines: We compare our model against several state-of-the-art methods: **FVS** [299] is an NVS method that uses reconstructed geometry (aggregated LiDAR in our implementation) as a "proxy" to re-project pixels from the input images into new camera views, where they are blended by a neural network. We enhance FVS to model dynamic actors. **Instant-NGP**[248] is a NeRF-based method that adopts multi-resolution hashing encoding for compact scene representation and efficient rendering. We enhance it by adding LiDAR depth supervision for better geometry and extrapolation. **NSG**[265] is a camera simulation method that models the scene with separate NeRF representations for the static background and each dynamic actor.

Controllability: As shown in Fig. 7-1, we can not only render the original scene, but because of our decomposed actor and background representations, we can also remove all the actors, and change their positions. With enhanced extrapolation capabilities, we can also change the SDV's location or test new sensor configurations.

Camera simulation: Sensor simulation should not only reconstruct nearby views, but also generate realistic data at significantly different viewpoints. Here we evaluate both settings. Similar to other NVS benchmarks [203], we subsample the sensor data by two, training on every other frame and testing on the remaining frames, dubbed *"interpolation"* test. We report PSNR, SSIM[377], and LPIPS[427]. We also evaluate extrapolation by simulating a new trajectory shifted laterally to the left or right by 2 or 3 meters, dubbed *"lane shift"* test. Since ground-truth is unavailable, we report the

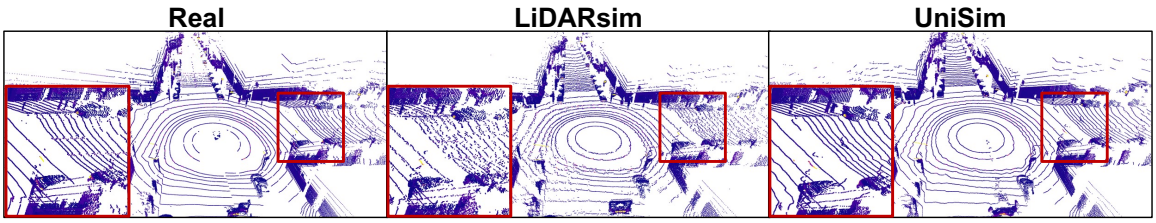


Figure 7-6: **Qualitative comparison of LiDAR simulation.** Our simulator generates LiDAR simulations with higher fidelity (*i.e.* less noise and more continuous beam rings). The simulated results are closer to real LiDAR data than those produced by [237].

FID[131] score. Due to computational costs of the baseline NSG, we select 10 scenes for evaluation.

As shown in Tab. 7.1, our method outperforms the baselines in all metrics, and the gap is more significant in extrapolation settings. FVS performs well on LPIPS and InstantNGP on PSNR in the interpolation setting, but both have difficulty when rendering at extrapolated views. Fig. 7-4 shows qualitative results. NSG produces decent results for dynamic actors but fails on large static scenes, due to its sparse multi-plane representation.

Ablation: We validate the effectiveness of several key components in Tab. 7.2. Both the actor model and the CNN decoder improve the overall performance over the neural features grid base model. The CNN is especially effective in the extrapolation setting, as it improves the overall image quality by spatial relation reasoning and increases model capacity. Adding perceptual and adversarial losses results in a small performance drop for interpolation, but improves the lane shift results.

LiDAR simulation: We also evaluate the fidelity of our LiDAR simulation and compare with SoTA approach LiDARSim [237]. For LiDARSim, we reconstruct surfel assets using all training frames, place actors in their original scenario in test frames, and perform ray-casting. Both methods use the real LiDAR rays to generate a simulated point cloud. We evaluate the fraction of real LiDAR points that have a corresponding simulated point (*i.e.*, Hit rate), the median per-ray ℓ_2 error and the average intensity simulation errors. As shown in Tab. 7.3, we outperform LiDARSim

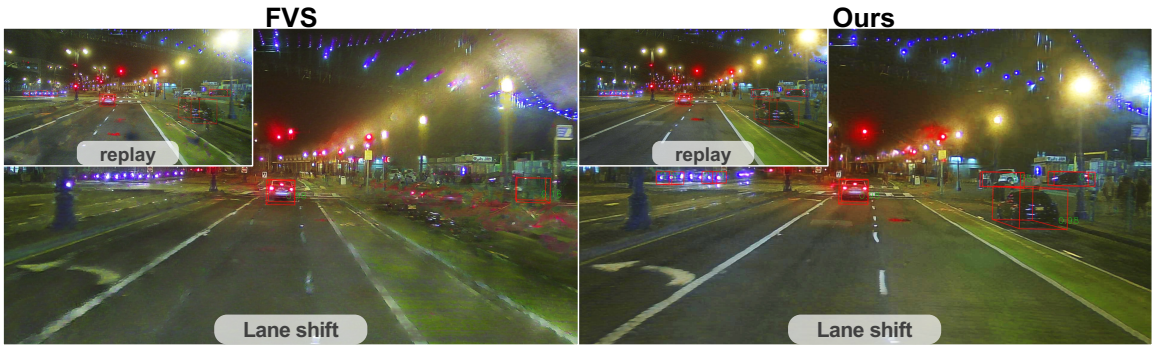


Figure 7-7: **Qualitative comparison of Real2Sim on replay and lane shift settings.**

	Log Replay		Lane Shift	
Method	Real2Sim	Sim2Real	Real2Sim	Sim2Real
FVS [299]	36.9	38.7	30.3	32.2
Instant-NGP [248]	22.6	34.0	18.1	26.5
Ours	**40.2**	**39.9**	**37.0**	**37.1**

Table 7.4: **Detection domain gap (mAP).** Real2Real = 40.9.

across all metrics, suggesting it is more accurate and has better coverage. Fig. 7-6 shows a visual comparison.

Domain gap in perception: In addition to image-similarity, sensor simulation should be realistic with respect to how autonomy perceives it. To verify if our simulation reduces the domain gap for perception tasks, we leveraged the SoTA camera-based birds-eye-view (BEV) detection model BEVFormer [200]. We consider two setups (a) **Real2Sim**: evaluating the perception model trained on real data on simulated data; (b) **Sim2Real**: training perception models with simulated data data and testing on real data. Specifically, we evaluate the real model on 24 simulated validation logs for Real2Sim and train perception models with 79 simulated training logs for Sim2Real. We consider both *replay* and *lane shift* test settings. In *replay*, we replay all actors and SDV with their original trajectories. In *lane shift*, we shift the SDV trajectory laterally by 2 meters and simulate images at extrapolated views. We report detection mean average precision (mAP).

As shown in Tab. 7.4, our approach achieves the smallest domain gap in both

	Instant-NGP [248]	FVS [299]	Ours
Sim	32.4	39.2	**41.4**
Real + Sim	40.1	41.1	**42.9**

Table 7.5: **Data augmentation via image simulation:** We train the detector on both simulated data and simulated + real data. With our sensor simulator, we can generate a large amount of synthetic yet realistic training data. We can also synthesize scenarios that do not exist (*e.g.* switching to the left lane or the right lane). This makes the detector more robust, and thus achieving better performance. Adding real world data back will further boost the performance. Real2Real = 40.9 mAP.

	Det. Agg. ↑	Pred. ADE ↓	Plan Cons. ↓
FVS [299]	0.80	2.35	6.15
Instant-NGP [248]	0.42	3.24	13.44
Ours	**0.82**	**1.68**	**6.09**

Table 7.6: **Open-Loop Real2Sim Autonomy evaluation**

Real2Sim and Sim2Real setups, on both *replay* and *lane shift* settings, while other existing approaches result in larger domain gaps, hindering their applicability to evaluate and train autonomy. This is especially evident in the more challenging *lane shift* setting, where there is a larger performance gap between our approach and the baselines. Fig. 7-7 shows the Real2Sim detection performance for both replay and lane shift settings compared to FVS [299].

Data augmentation with simulation data: Next, we study if our simulated data boosts performance when used for training. Specifically, we use all PandaSet training logs to generate simulation variations (replay, lane shift 0.5 and 2 meters) to train the detectors. As shown in Tab. 7.5, using only our simulated data to train the perception model is even better than training with all real data. Note we only increase the rendered viewpoints and do not alter the content. We then combine the real data with the simulation data and retrain the detector. Tab. 7.5 shows that augmentation yields a significant performance gain. In contrast, baseline data augmentation brings marginal gain or harms performance.

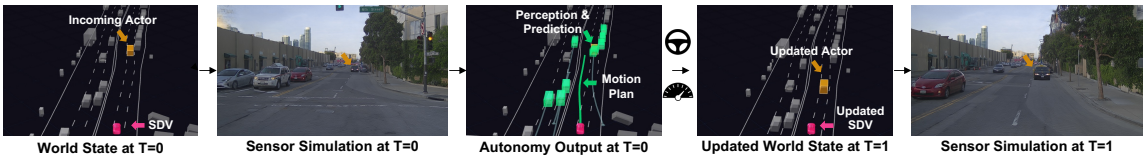


Figure 7-8: **Closed-loop evaluation**. From left to right: we create a safety-critical scenario (e.g., incoming actor), simulate the sensor data, run autonomy on it, update the SDV's viewpoint and other actor locations, and simulate the new sensor data.

Domain gap in prediction and planning: Sensor simulation not only affects perception tasks, but also downstream tasks such as motion forecasting and planning. We report domain gap metrics by evaluating an autonomy system trained on real data on simulated images of the original scenario. The autonomy system under evaluation is a module-based system, with BEVFormer [200] taking front-view camera images as input and producing BEV detections that are matched over time to produce tracks via greedy association as the perception module. These are then fed to a motion forecasting model [67] that takes in BEV tracks and a map raster and outputs bounding boxes and 6 second trajectory forecasts. Finally a SoTA sampling-based motion planner [306] takes the prediction output and map to plan a maneuver. We report open-loop autonomy metrics (detection agreement @ IoU 0.3, prediction average displacement error (ADE), and motion plan consistency at 5 seconds) in Table 7.6. Compared to other methods, our approach has the smallest domain gap. Please see supp. for details.

Closed-loop simulation: With our simulation system, we can create new scenarios, simulate the sensor data, run the autonomy system, update the state of the actors in a reactive manner and the SDV's location, and execute the next time step (see Fig. 7-8). This gives us a more accurate measure of the SDV's performance to how it would behave in the real world for the same scenario. Fig. 7-1 shows additional simulations of the autonomy on safety critical scenarios such as an actor cutting into our lane or an oncoming actor in our lane. The SDV then lane changes, and with our simulator we can simulate the sensor data realistically throughout the scenario. Please see supp. video for complete visuals.

7.5.2 LiDAR Scene Generation

We first compare our generation results on KITTI-360 [202] with other baselines under the unconditional generation setting. Following LiDARGen [442], we use the first two sequences as the validation set and exploit the rest for model training. Fig. 7-5[*] shows a few qualitative results of our model and the two baselines. We also include the real data for reference. Our generated results are much more similar to the real data, with more structured and reasonable scene layouts and more stable/clearer beam patterns. More unconditional generation results on KITTI-360 can be found in the second row of Fig. 7-2 and supp. materials.

Quantitative results: Following [442], we use the Maximum-Mean Discrepancy (MMD) and Jensen–Shannon divergence (JSD) with a 100×100 2D histogram along the ground plane (x and y coordinates) as metrics. Since our model generates points based on voxels, the number of points may differ from the real point cloud. We thus use occupancy as a measurement when doing histogram bin count (*i.e.*, points from the same voxel will only count once) for point clouds from real data and other baselines. We believe this change captures the global structure difference better and lowers the weight on the local point density estimation, which is more reasonable and aligns better with the perceptual quality. As shown in Tab. 7.7, our method achieves superior performance compared with the baselines.

Model parameters: We calculate the number of parameters to make sure the model capacities are at the same level when compared with baselines. The number of parameters of LiDARGen [442] and our model are 29.7M and 40.3M, which are both smaller than a standard Swin-Small model.

Human study: We perform an A/B test on a set of 8 researchers who have LiDAR expertise to better evaluate the visual quality of the generated samples. We use the same test system as [442], which shows a pair of randomly chosen images of two point

[*]We contacted the authors of [442] to obtain outputs of all models they used for visualization and metrics calculation

Method	MMD_{BEV}	JSD_{BEV}
LiDAR VAE	1.18×10^{-3}	0.256
LiDAR GAN	2.07×10^{-3}	0.275
Projected GAN	1.18×10^{-2}	0.332
LiDARGen	4.80×10^{-4}	0.140
Ours	9.67×10^{-5}	0.132

Table 7.7: Quantitative comparison of LiDAR generation.

Comparison	Preference rate
Ours vs LiDAR VAE [40]	99.5%
Ours vs LiDAR GAN [40]	100%
Ours vs ProjGAN [312]	100%
Ours vs LiDARGen [442]	98.5%

Table 7.8: **Human evaluation on LiDAR generation.** Results from our model show significantly better visual quality.

clouds each time and lets the human decide which one is more realistic. We compare with four baselines as well as with the real data, with 200 image pairs each, leading to 1000 image pairs in total. We show the percentage of examples where participants believed our generations are more realistic against other baselines in Tab. 7.8. It is clear that our model can generate results with superior visual quality; over 98.5% of the time (100% for some baselines), the testers prefer our results over the baselines. It is worth noting that in 32% cases, people believe our results are *more realistic than real data*, which is very significant given the fact that for data that is indistinguishable from real, the winning ratio would be 50% with random choice.

Unconditional generation: Besides the KITTI-360 results in Fig. 7-5 and Fig. 7-2, we also train our model on dense Panadset for dense point cloud generation; and additionally run detection models on the generated samples, shown in Fig. 7-9. We can see that our model is able to generate diverse scenes with proper actor placement (*e.g.*, parked car in the right example), indicating the superiority of our model for this LiDAR generation task. Moreover, the excellent detection results showcase that the generated samples also have high fidelity w.r.t. the perception model.

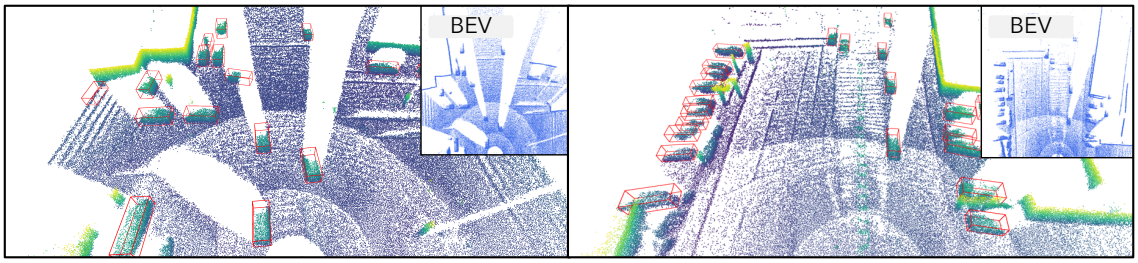


Figure 7-9: **Unconditional LiDAR generation on Pandaset.** We train our model on dense Pandaset data and generate dense results. The generated samples show diverse scenario layouts with proper actor placement (*e.g.*, parked car in the right sample). The synthesized point clouds are realistic such that a pre-trained detector can directly work out-of-the-box.

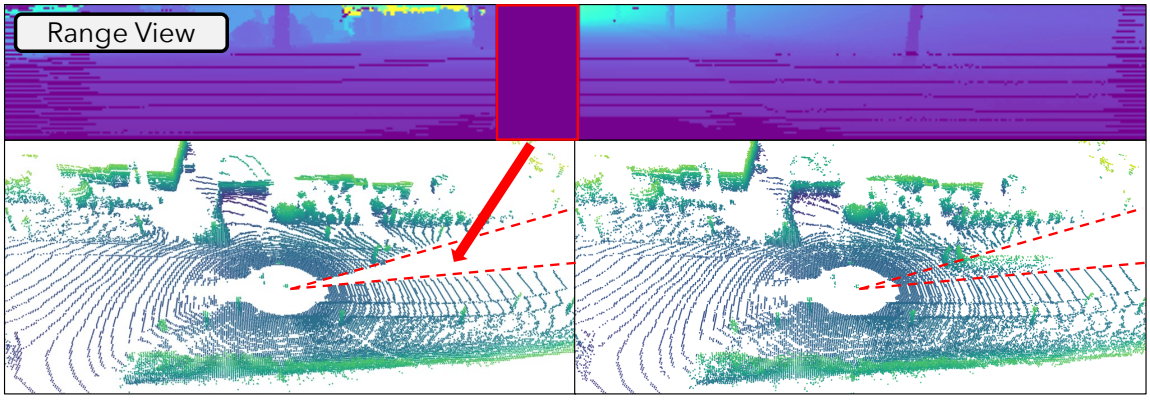


Figure 7-10: **Conditional LiDAR generation for dirt removal.** We mask the red rectangular region in the range view image to mimic dirt occluder. **Left**: Original input. The masked region is not visible to the model. **Right:** Our generation results. Our model can successfully recover the vehicle that is partially observed.

Conditional generation: We show conditional generation results on KITTI-360 in the bottom row of Fig. 7-2. Our model can fully exploit the visible part(colored by purple) as context, do reasonable extrapolation for the surrounding environment, and generate diverse scenes (*e.g.*, curved road or crossroad) that align with the input condition well. See supp. materials for more results on KITTI-360/PandaSet.

Meanwhile, we further consider a practical setting where the LiDAR sensor can fail for a specific region due to dirt or mechanical issues. To mimic this situation, we mask a part of the range images, as shown in Fig. 7-10. We can see that our model can still do accurate completion even on partially visible cars, recovering the occluded region, and potentially avoiding dangerous situations.

Manipulation: We show manipulation results with actor insertion and removal in the second row of Fig. 7-2. This is achieved by explicitly changing the code indices on the code map and letting the decoder generate new results. For example, we can copy the codes for the ground plane and paste them to overwrite the region where a car exists, resulting in a controllable manipulation process. We refer the readers for more results in supp. materials.

Chapter 8

Epilogue

In recent few years, we have witnessed amazing progress on various 3D applications, from self-driving vehicles to industrial robots, and from digital editing to AR/VR systems. One can totally imagine the bright future in front of us when all these systems have been deployed into our lives and transformed the way we live and the way we think. However, the future is not here yet. Existing 3D systems, while powerful and effective, may start to fail and break apart when extending to challenging unconstrained real-world scenarios. One key reason is that they lack structural understanding of our world.

In this dissertation, we focus on addressing the following two questions: (i) how to robustly model and reason about the visible world that we see, and (ii) how to hallucinate the unseen and imagine novel scenarios in a realistic fashion? We present a set of robust computational tools that allow one to effectively model and simulate the dynamic 3D world from real-world unconstrained sensory data. By infusing different structures and prior knowledge of the world into the systems, we are able to design machines that can not only understand what they see, but also imagine beyond. Specifically, we first focused on developing robust computational models that can leverage physical and geometric structures to effectively model and reason about what we see. Then, we pushed the limit of existing 3D modeling algorithms. By teaching these models how to hallucinate beyond what they see, we are able to make those modeling algorithms work under scenarios that they couldn't before. Finally, we want

to not just model what happened in the world, but also be able to hallucinate and imagine all sorts of "what-if" counterfactual scenarios. We thus developed several techniques that help us build an editable virtual copy of the world such that we can re-compose them for various simulation and generation.

8.1 Future Directions

Knowledge-grounded reasoning and modeling: Humans are generalists. We can reason about various 3D quantities based on different combinations of cues and quickly adapt to new environments. Most existing data-driven intelligent systems, however, are specialists. They are trained to excel at one or a few tasks or setups. When given similar tasks in new settings or encountering slightly different data distributions, they may fail catastrophically, since they cannot reliably transfer their understandings. Thus, it is critical to ground the reasoning and modeling process with various knowledge so that the systems can be more robust and applicable to various scenarios. One possibility is to equip the systems with the ability to automatically select and adapt structured scene representations for different tasks and under various setups. This is important since different representations have different properties and pros and cons. In this disseration, we have showcased how to distill knowledge of objects into structure from motion algorithms such that they can adaptively select different types of correspondences to reconstruct scenes under different scenarios. This can serve as a strong starting point. Second, data from different modalities are complementary and encode different information. For instance, thermal images encode traces of past human-object interactions which are not visible in visual domain. It is important to develop methods that integrate knowledge across modalities to better model our world. Finally, intelligent systems should be able to quickly adapt to different environments and scenarios with little or even no supervision. To achieve this, the models must capture the relationships among data and transfer the knowledge and structure across tasks. Recent advances in large language models have showcased great potential in this direction. Building upon their success and develop foundation

models for 3D vision will potentially allow for a new paradigm in the design and testing of various 3D tasks with unparalleled scalability.

Digital twins for all: The future of digital twins holds great promise. By realistically simulating motion, physics, and materials, they can stand to impact different facets of our lives, such as robot autonomy, AR/VR, transportation, and mobility. The works presented in this thesis are just the start of an exciting chapter. To fully unlock the potential of simulation and generation, one natural yet non-trivial step is to extend current efforts to 4D (*i.e.*, 3D + time) and simulate the movement of agents and the interactions among them. This is of crucial importance since autonomous agents will ultimately operate in the vicinity of one another or humans. They must learn to anticipate the intentions or the outcome of actions so that they can respond in a socially compliant way. Furthermore, human body motion and gestures are non-verbal communication tools indicative of their thoughts. To build a Metaverse where people can interact effectively, one must capture the scocial dynamics, which drives the interactions among humans and agents. Besides socially-driven motions, integrating physical dynamics into the virtual world is also critical for realistic motion simulation of inanimate objects. For instance, by incorporating a physics-based link model from the field of botany, one can potentially simulate how tree branches vibrate. This will allow one to create a more vivid virtual experience and potentially produce positive impacts on other fields — by better modeling the 3D geometry of the trees in an urban city, we can better estimate the cooling effects they bring and improve the urban ecosystem. Going beyond AR/VR and robot simulation, another interesting direction is how digital twins can improve urban transportation and planning. One major challenge in transportation is identifying and quickly responding to sudden commuting needs, traffic, and congestion. By building a virtual copy of the city, we can enable planners to test every possible scenario, forecast the potential outage, optimize the resource, and answer questions such as where to add bike lanes and how to adjust the traffic signal time. Finally, existing efforts on ecosystem monitoring mostly focus on 2D modeling and recognition. Simulating (endangered) animals and

plants in 3D can help us understand them more, and the 3D digital captures can be beneficial when educating and engaging younger generations about endangered species.

Appendix A

Supplementary: Sparse Semantic Localization

A.1 Quantitative Analysis

Error vs Travelling Distance Fig. A-1 depicts the localization error as a function of travelling distance. For this figure we can see that our approach is relatively stable across travelling distance without catastrophic failures. Particularly, median lateral error is maintained to be around 5cm with worst case around 23cm.

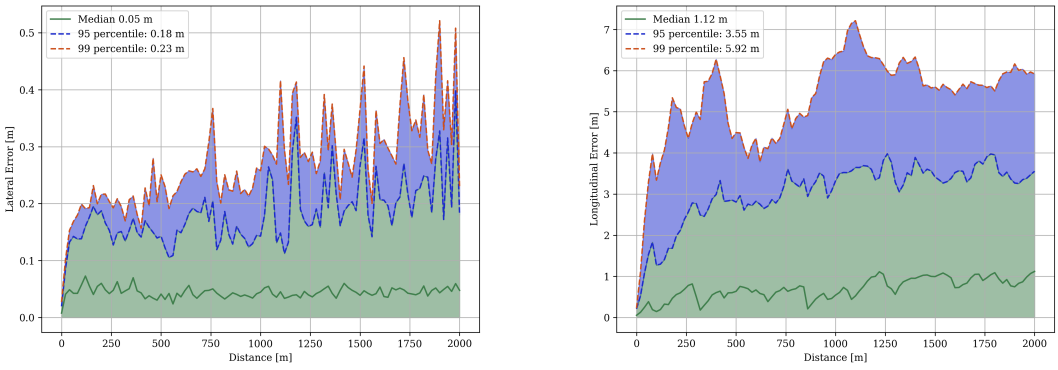


Figure A-1: Localization error as a function of travelling distance.

A.2 Additional Results on Hyper-Parameter Search

Hyper-parameter Search Procedure We performed an exhaustive grid search to find the set of hyperparameters that minimize the failure rate in the mini-validation set. The search grid consisted of soft-argmax temperature from the set $\{1., 8.\}$, lane temperature from $\{.5, 1., 1.5\}$, sign temperature from $\{.5, 1., 1.5\}$, GPS observation's standard deviation at $\{0, 10, 20\}$ meters, dynamics observation's standard deviation from $\{1., 2.\}$ degrees, dynamics observation's longitudinal standard deviation from $\{5., 10.\}$ meters. This describes a grid of 216 points for which metrics were computed exhaustively the best set of hyper-parameters can be seen in Tab. A-2.

Method	GPS std	Dyn Angle std	Dyn Lat std	Dyn Lon std	Argmax Temp	Lane Temp	Sign Temp
Lane	-	1.	2.	5.	1.	1.	-
Lane+GPS	20	1.	2.	5.	1.	1.	-
Lane+Sign	-	1.	2.	5.	1.	1.	2.5
All	20	1.	2.	5.	1.	1.	2.5

Figure A-2: Best hyper-parameters for each method

Hyper-parameter Search Results We showcase each hyper-parameter's results in Fig. A-3. From this figure we can see how the choice of each term influences different metrics. For instance, trusting GPS more in general hurts the performance lateral localization but does not have a significant influence on longitudinal. This is due to the lane detection cue as a centi-meter level source for lateral error. Moreover, overall a soft-argmax that is more close to 'mean' is overall better than the one that is more close to 'arg-max'. Another interesting findings are that assuming a relative large vehicle dynamics noise in both lateral and longitudinal direction is in practice better. Moreover, we find a high temperature is needed for the sign energy, such that as soon as it appears, it can help close loop and reduce longitudinal drift.

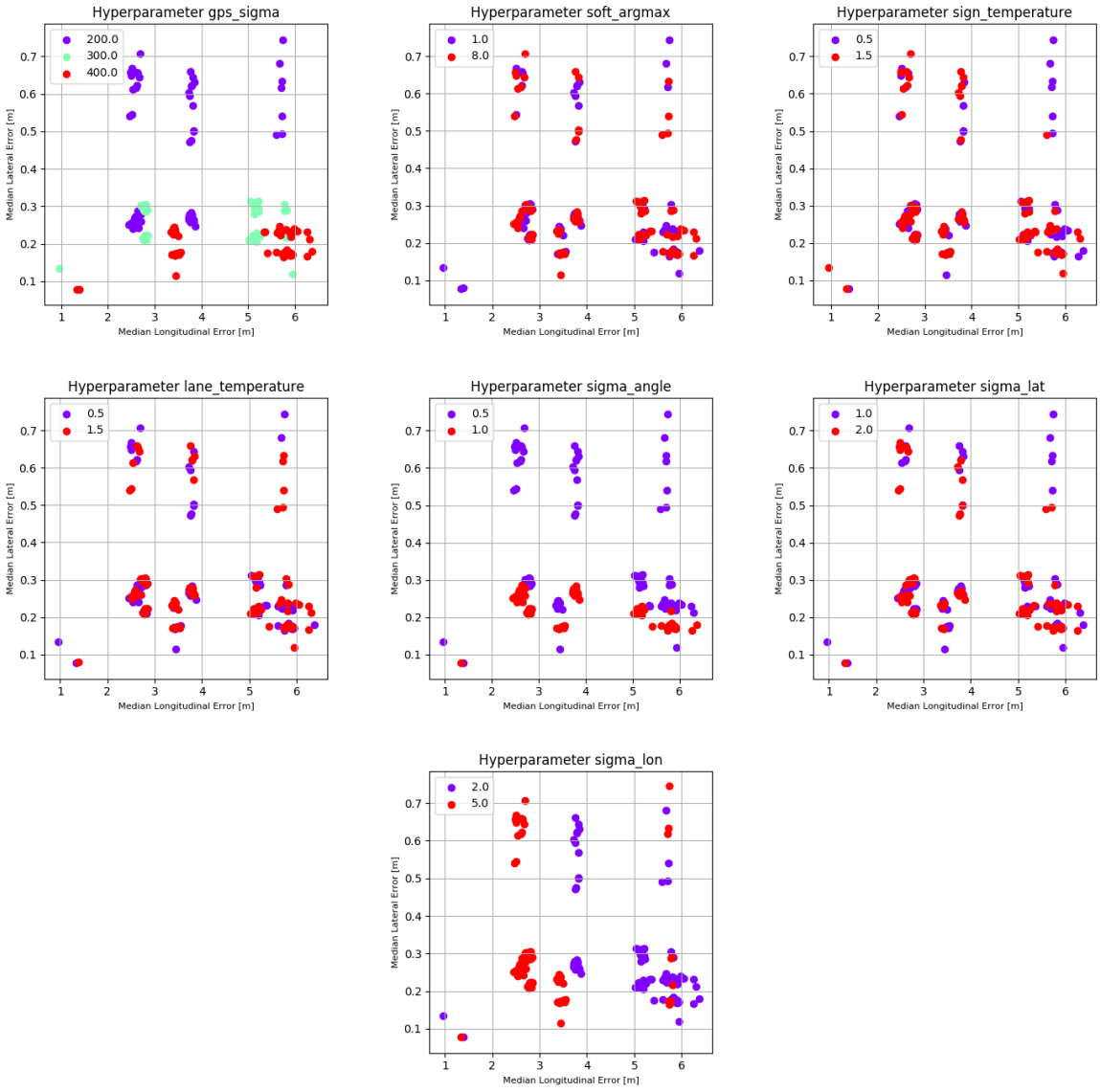


Figure A-3: Grid search result per each hyper-parameter.

Appendix B

Supplementary: Deep Structured Motion Estimation

B.1 Derivation of the Solver

In this section, we first derive the Jacobian of the energy function in a general form. Then we describe the specific formulation for each term.

B.1.1 Gauss-Newton Solver

The proposed energy terms can all be expressed in the following form:

$$E(\boldsymbol{\xi}; \mathcal{I}) = \sum_{\boldsymbol{p}} \rho(r(\boldsymbol{p}, \boldsymbol{\xi}; \mathcal{I})),$$

where $\boldsymbol{p}$ refers to the pixels belonging to an instance, ρ is a robust penalty function, and r is a residual function. The goal is to find a transformation $\boldsymbol{\xi}$ that minimizes the energy, or equivalently the sum of square residuals:

$$\boldsymbol{\xi} = \arg\min_{\boldsymbol{\xi}} E(\boldsymbol{\xi}; \mathcal{I}) = \arg\min_{\boldsymbol{\xi}} \sum_{\boldsymbol{p}} \rho(r(\boldsymbol{p}, \boldsymbol{\xi}; \mathcal{I})).$$

As mentioned in the paper, we use the same ρ for all energy terms where $\rho(x) = \tau(x^2) = (x^2 + \epsilon^2)^\alpha$.

As the residual function is non-linear and cannot be solved exactly, we iteratively approximate it with a first order Taylor series expansion and search for the local minimum. More formally, the residual function can be approximated as

$$r'(\boldsymbol{p}, \Delta\boldsymbol{\xi}) = r(\boldsymbol{p}, \boldsymbol{\xi} + \Delta\boldsymbol{\xi}; \mathcal{I}) \approx r(\boldsymbol{p}, \boldsymbol{\xi}; \mathcal{I}) + \frac{\partial r(\boldsymbol{p}, \boldsymbol{\xi}; \mathcal{I})}{\partial \boldsymbol{\xi}} \Delta\boldsymbol{\xi}.$$

For convenience, we define the Jacobian $\frac{\partial r(\boldsymbol{p}, \boldsymbol{\xi}; \mathcal{I})}{\partial \boldsymbol{\xi}}$ as $\mathbf{J}_{\boldsymbol{p}}$. Thus we can solve the following surrogate optimization problem instead:

$$\begin{aligned}
\Delta\boldsymbol{\xi}^* &= \arg\min_{\Delta\boldsymbol{\xi}} \sum_{\boldsymbol{p}} \rho(r'(\boldsymbol{p}, \Delta\boldsymbol{\xi})) \\
&= \arg\min_{\Delta\boldsymbol{\xi}} \sum_{\boldsymbol{p}} \tau(r'(\boldsymbol{p}, \Delta\boldsymbol{\xi})^T r'(\boldsymbol{p}, \Delta\boldsymbol{\xi})) \\
&= \arg\min_{\Delta\boldsymbol{\xi}} \sum_{\boldsymbol{p}} \tau((r(\boldsymbol{p}, \boldsymbol{\xi}) + \mathbf{J}_{\boldsymbol{p}}\Delta\boldsymbol{\xi})^T (r(\boldsymbol{p}, \boldsymbol{\xi}) + \mathbf{J}_{\boldsymbol{p}}\Delta\boldsymbol{\xi}))
\end{aligned}$$

where we simply denote $\rho(x) = \tau(x^2)$. The minimum happens at the point where the gradient is equal to zero. Let us denote $L_{\boldsymbol{p}} = r'(\boldsymbol{p}, \Delta\boldsymbol{\xi})^T r'(\boldsymbol{p}, \Delta\boldsymbol{\xi})$ and $r'_{\boldsymbol{p}} = r'(\boldsymbol{p}, \Delta\boldsymbol{\xi})$. Thus we have:

$$\begin{aligned}
0 &= \frac{\delta E}{\delta \Delta\boldsymbol{\xi}} \\
0 &= \sum_{\boldsymbol{p}} \frac{\delta \tau(L_{\boldsymbol{p}})}{\delta L_{\boldsymbol{p}}} \frac{\delta L_{\boldsymbol{p}}}{\delta r'_{\boldsymbol{p}}} \frac{\delta r'_{\boldsymbol{p}}}{\delta \Delta\boldsymbol{\xi}} \\
0 &= \sum_{\boldsymbol{p}} \frac{\delta \tau(L_{\boldsymbol{p}})}{\delta L_{\boldsymbol{p}}} \cdot 2(r'_{\boldsymbol{p}})^T \cdot \mathbf{J}_{\boldsymbol{p}} \\
0 &= \sum_{\boldsymbol{p}} \frac{\delta \tau(L_{\boldsymbol{p}})}{\delta L_{\boldsymbol{p}}} \cdot 2(r(\boldsymbol{p}, \boldsymbol{\xi}) + \mathbf{J}_{\boldsymbol{p}}\Delta\boldsymbol{\xi})^T \cdot \mathbf{J}_{\boldsymbol{p}}
\end{aligned}$$

Since the equation now takes a linear form w.r.t. $\Delta\boldsymbol{\xi}$, we can solve exactly in close

form. In particular, the minimum occurs when

$$\sum_{\boldsymbol{p}} \frac{\delta\tau(L_{\boldsymbol{p}})}{\delta L_{\boldsymbol{p}}} \mathbf{J}_{\boldsymbol{p}}^T \mathbf{J}_{\boldsymbol{p}} \Delta\boldsymbol{\xi} = -\sum_{\boldsymbol{p}} \frac{\delta\tau(L_{\boldsymbol{p}})}{\delta L_{\boldsymbol{p}}} \mathbf{J}_{\boldsymbol{p}}^T r(\boldsymbol{p}, \boldsymbol{\xi}; \mathcal{I}),$$

$$\Delta\boldsymbol{\xi} = -(\sum_{\boldsymbol{p}} \mathbf{J}_{\boldsymbol{p}}^T \mathbf{W}_{\boldsymbol{p}} \mathbf{J}_{\boldsymbol{p}})^{(-1)} \sum_{\boldsymbol{p}} \mathbf{J}_{\boldsymbol{p}}^T \mathbf{W}_{\boldsymbol{p}} r(\boldsymbol{p}, \boldsymbol{\xi}; \mathcal{I}),$$

where $\mathbf{W}$ is a diagonal matrix with diagonal entries equal to $\frac{\delta\tau(L_{\boldsymbol{p}})}{\delta L_{\boldsymbol{p}}}$. We hence take a step and update the transformation $\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} \circ \Delta\boldsymbol{\xi}$.

We next describe the residual function and the Jacobian for each term. We use the same notations as in the paper. We further define $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T = \boldsymbol{\xi} \circ \pi_{\mathbf{K}}^{-1}(\boldsymbol{p}, \mathcal{D}(\boldsymbol{p}))$ as the 3D coordinate of the pixel $\boldsymbol{p}$ after inverse depth warping and applying rigid transform $\boldsymbol{\xi}$.

B.1.2 Jacobian of Each Energy Term

Photometric error: The residual function of the photometric error is simply the RGB pixel value difference. Formally, it is defined as:

$$r(\boldsymbol{p}, \boldsymbol{\xi}; \mathcal{I}) = \mathcal{L}^1(\boldsymbol{p}') - \mathcal{L}^0(\boldsymbol{p}).$$

The Jacobian is defined as:

$$\begin{aligned}
\mathbf{J} &= \frac{\partial r(\boldsymbol{p}, \boldsymbol{\xi}; \mathcal{I})}{\partial \boldsymbol{\xi}} = \frac{\partial \mathcal{L}^1(\boldsymbol{p}') - \mathcal{L}^0(\boldsymbol{p})}{\partial \boldsymbol{\xi}} \\
&= \frac{\partial \mathcal{L}^1(\boldsymbol{p}')}{\partial \boldsymbol{p}'} \frac{\partial \pi_{\mathbf{K}}(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \\
&= \begin{bmatrix} \nabla_x \mathcal{L}^1 \\ \nabla_y \mathcal{L}^1 \end{bmatrix} \begin{bmatrix} \frac{f_x}{z} & 0 & -\frac{x f_x}{z^2} \\ 0 & \frac{f_y}{z} & -\frac{y f_y}{z^2} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & z & -y \\ 0 & 1 & 0 & -z & 0 & x \\ 0 & 0 & 1 & y & -x & 0 \end{bmatrix}
\end{aligned}$$

Rigid fitting: The residual function of the rigid fitting term is simply the 3D distance between the 3D point and its correspondence. Formally, it is defined as:

$$\begin{aligned} r(\boldsymbol{p}, \boldsymbol{\xi}; \mathcal{I}) &= \boldsymbol{\xi} \circ \pi_{\mathbf{K}}^{-1}\left(\boldsymbol{p}, \mathcal{D}^0(\boldsymbol{p})\right) - \pi_{\mathbf{K}}^{-1}\left(\boldsymbol{q}, \mathcal{D}^1(\boldsymbol{q})\right) \\ &= \mathbf{x} - \pi_{\mathbf{K}}^{-1}\left(\boldsymbol{q}, \mathcal{D}^1(\boldsymbol{q})\right). \end{aligned}$$

The Jacobian is thus:

$$\begin{aligned} \mathbf{J} = \frac{\partial r(\boldsymbol{p}, \boldsymbol{\xi}; \mathcal{I})}{\partial \boldsymbol{\xi}} &= \frac{\partial \mathbf{x} - \pi_{\mathbf{K}}^{-1}\left(\boldsymbol{q}, \mathcal{D}^1(\boldsymbol{q})\right)}{\partial \boldsymbol{\xi}} \\ = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} &= \begin{bmatrix} 1 & 0 & 0 & 0 & z & -y \\ 0 & 1 & 0 & -z & 0 & x \\ 0 & 0 & 1 & y & -x & 0 \end{bmatrix} \end{aligned}$$

Flow consistency: The residual function of the flow consistency term is simply the difference between the estimated flow and the projection of the estimated 3D motion. Formally, it is defined as:

$$r(\boldsymbol{p}, \boldsymbol{\xi}; \mathcal{I}) = (\boldsymbol{p}' - \boldsymbol{p}) - \mathcal{F}_{\mathcal{L}}(\boldsymbol{p}).$$

The Jacobian is defined as:

$$\begin{aligned} \mathbf{J} &= \frac{\partial (\boldsymbol{p}' - \boldsymbol{p}) - \mathcal{F}_{\mathcal{L}}(\boldsymbol{p})}{\partial \boldsymbol{\xi}} = \frac{\partial \boldsymbol{p}'}{\partial \boldsymbol{\xi}} = \frac{\partial \boldsymbol{p}'}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \\ &= \frac{\partial \pi_{\mathbf{K}}(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \\ &= \begin{bmatrix} \frac{f_x}{z} & 0 & -\frac{x f_x}{z^2} \\ 0 & \frac{f_y}{z} & -\frac{y f_y}{z^2} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & z & -y \\ 0 & 1 & 0 & -z & 0 & x \\ 0 & 0 & 1 & y & -x & 0 \end{bmatrix} \end{aligned}$$

B.2 Impact of Unrolling Steps in GN Solver

In this section, we study the trade-off between performance and runtime with respect to the maximum number of unrolling steps in the GN solver. As shown in Fig. B-1, our

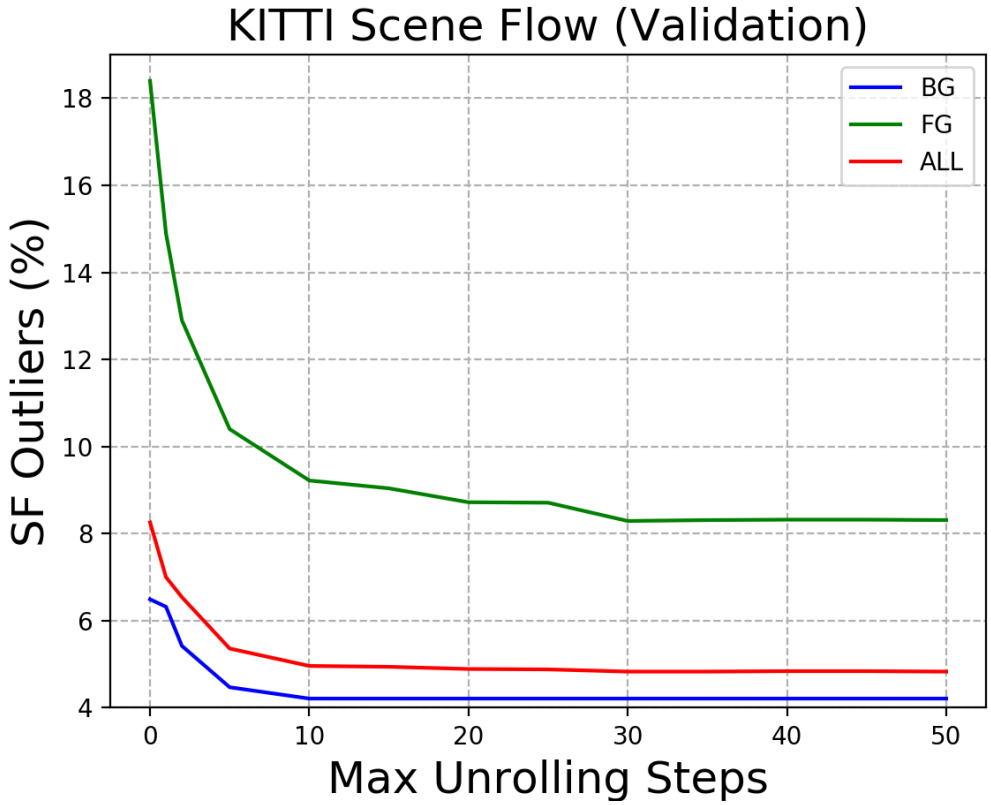


Figure B-1: **Performance vs unrolling steps.** While foreground requires 30 steps to converge, background achieves best results within 10 steps. As a consequence, the overall performance reaches plateau after 10 steps, with only minor improvement. We note that despite the foreground outliers reduce quite a bit after 10 steps, it is mainly caused by a few instances. Most instances converges within 10 steps.

model can achieve very good performance within 10 steps. The overall performance still improves a bit as the optimization procedure proceeds, since a few foreground instances require longer time to converge. In practice, many instances converge within 10 steps, which leads to the speed boost in Fig. B-2. To gain more intuition, we also visualize the scene flow error map at different iterations in Fig. B-3. For more qualitative results, we refer the readers to the attached video.

B.3 Impact of energy functions

To understand the effectiveness of each energy term on background and foreground objects, we evaluate our model with different energy combinations. The full ablation

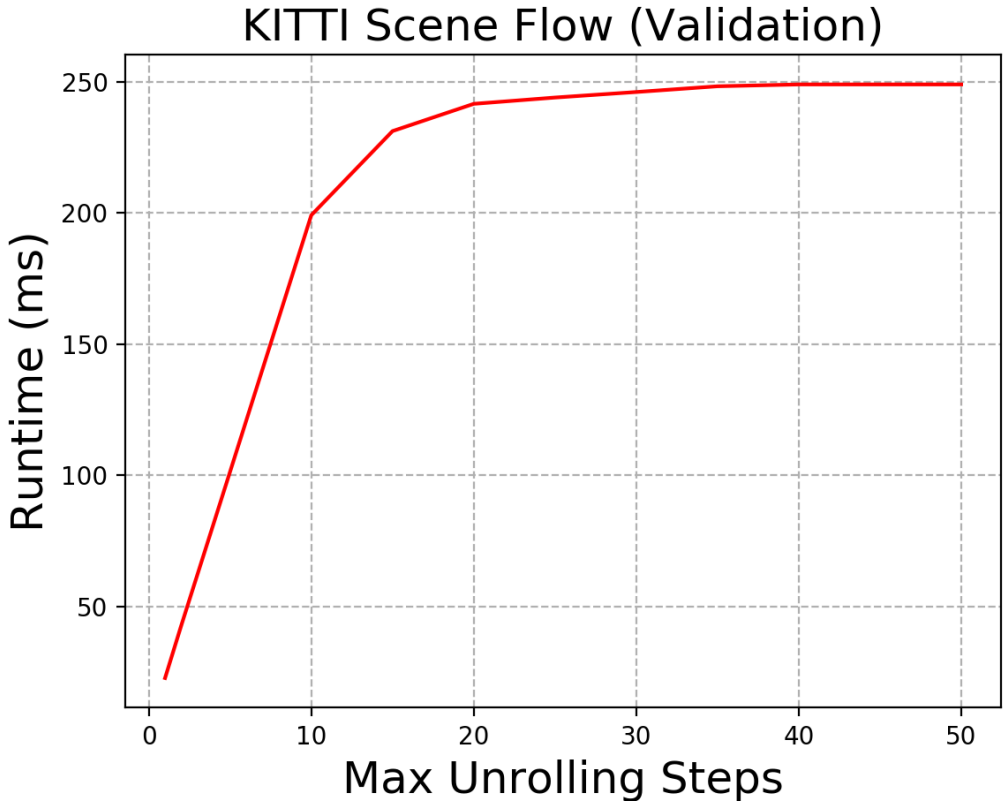


Figure B-2: **Runtime vs unrolling steps.** At first the runtime scales linearly with the maximum number of steps. As more instances converge (*i.e.* the energy reaches plateau) and terminate, the runtime becomes faster.

table is shown in Tab. B.1. While best performance is achieved for foreground objects when using all energy terms, the error is lowest for background when employing only photometric term. This can be explained by the fact that vehicles are often texture-less, and sometimes have large displacements. If we only employ photometric term, it will be very difficult to establish correspondences and handle drastic appearances changes. With the help of flow and rigid term, we can guide the motion and reduce such effect, and deal with occlusions. In contrast, background is full of discriminative textures and has relatively small motion, which is ideal for photometric term. Adding other terms may introduce extra noise and degrade the performance.

Employed energy			Background outliers (%)			
E_{pho}	E_{flow}	E_{rigid}	D1	D2	Fl	SF
✓			1.92	2.69	**3.71**	**4.30**
	✓		1.92	2.57	4.73	5.31
		✓	1.92	2.59	5.92	6.45
✓	✓		1.92	2.57	4.25	4.82
	✓	✓	1.92	2.56	4.72	5.28
✓		✓	1.92	**2.55**	4.69	5.25
✓	✓	✓	1.92	2.56	4.63	5.21

Employed energy			Foreground outliers (%)			
E_{pho}	E_{flow}	E_{rigid}	D1	D2	Fl	SF
✓			1.70	4.25	7.57	9.00
	✓		1.70	5.53	8.39	10.5
		✓	1.70	3.72	14.9	16.1
✓	✓		1.70	5.18	7.97	9.85
	✓	✓	1.70	4.58	6.98	8.67
✓		✓	1.70	**3.70**	7.98	9.24
✓	✓	✓	1.70	4.56	**6.73**	**8.39**

Table B.1: **Contributions of each energy:** As foreground objects sometimes are texture-less and have large displacement, simple photometric term is not enough. In contrast, background is full of discriminative cues. Simple photometric error would suffice. Adding extra terms will introduce noises and hurt the performance.

B.4 Curating KITTI Scene Flow

While creating the instance-wise 3D rigid motion ground truth (GT) from KITTI scene flow dataset, we discover two critical issues: firstly, there are mis-alignments over instance boundaries between the GT scene flow and GT instance segmentation; secondly, we find that the scale of the same 3D instance changes across two frames, which is impossible in practice.

Fig. B-4 shows two examples where the GT scene flow and GT segmentation mis-align. For each pixel, we determine its scene flow based instance label by finding if its scene flow better fits the rigid motion of a foreground instance or the background scene. We also get another source of labels from instance segmentation task. If a point is defined as foreground instance by both scene flow and the instance segmentation, it is colored in yellow. It is colored in green/red if only the scene flow/segmentation indicates it belongs to foreground object. If both agree it is background, we colored

it in blue. In general, the mis-alignment happens at the boundaries. It is expected as the amount of CAD models (employed to compute GT flow/disparity) and their underlying transformations are limited and it is difficult to cover all types of cars on the road while GT instance are labeled by humans through polygons. The mis-alignment will also result in incorrect rigid motion*, especially when there are many outliers. We thus *re-label* the points and compute the rigid motion again. This greatly reduces the outliers ratio.

Even after fixing the instance boundary mis-alignment issue, we find it is still difficult to fit a rigid transform for a few instances that makes GT flow and GT disparity agree with each other. We suspect this is due to the fact that some instances do not undergo rigid transform in the data. To solidify our claim, we further compute the 3D distance between the same pair of points from an instance at different time step. To be more specific, we obtain the correspondences at different frames using GT flow and exploit GT disparity to compute their respective 3D coordinates. As shown in Fig. B-5, the 3D distance between the points changes quite a bit for the vehicle. It seems like the underlying transformation across the frames is not rigid. We verify with the authors [243] and they confirm that an additional scale parameter is employed to fit the 3D CAD model to each frame independently during the GT creation stage. Several objects thus do not undergo a rigid transform. To address this, we simply treat them as don't care regions, and ignore them when computing our metrics.

B.5 Qualitative Results

Fig. B-6 and Fig. B-7 provide more qualitative comparison against the baselines. Fig. B-8 and Fig. B-9 show a few examples where our model fails.

*As there are no GT for the rigid motion, we evaluate its quality using the scene flow metric. To be more specific, we use the GT D1 and the fitted rigid motion to compute scene flow via Eq. 6 and measure the number of outliers.

B.6 KITTI Scene Flow Benchmark

Fig. B-10 shows the screenshot of the KITTI scene flow leaderboard at the time of paper submission. Our method outperforms all previous methods, including anonymous submissions, by a significant margin in both runtime and performance.

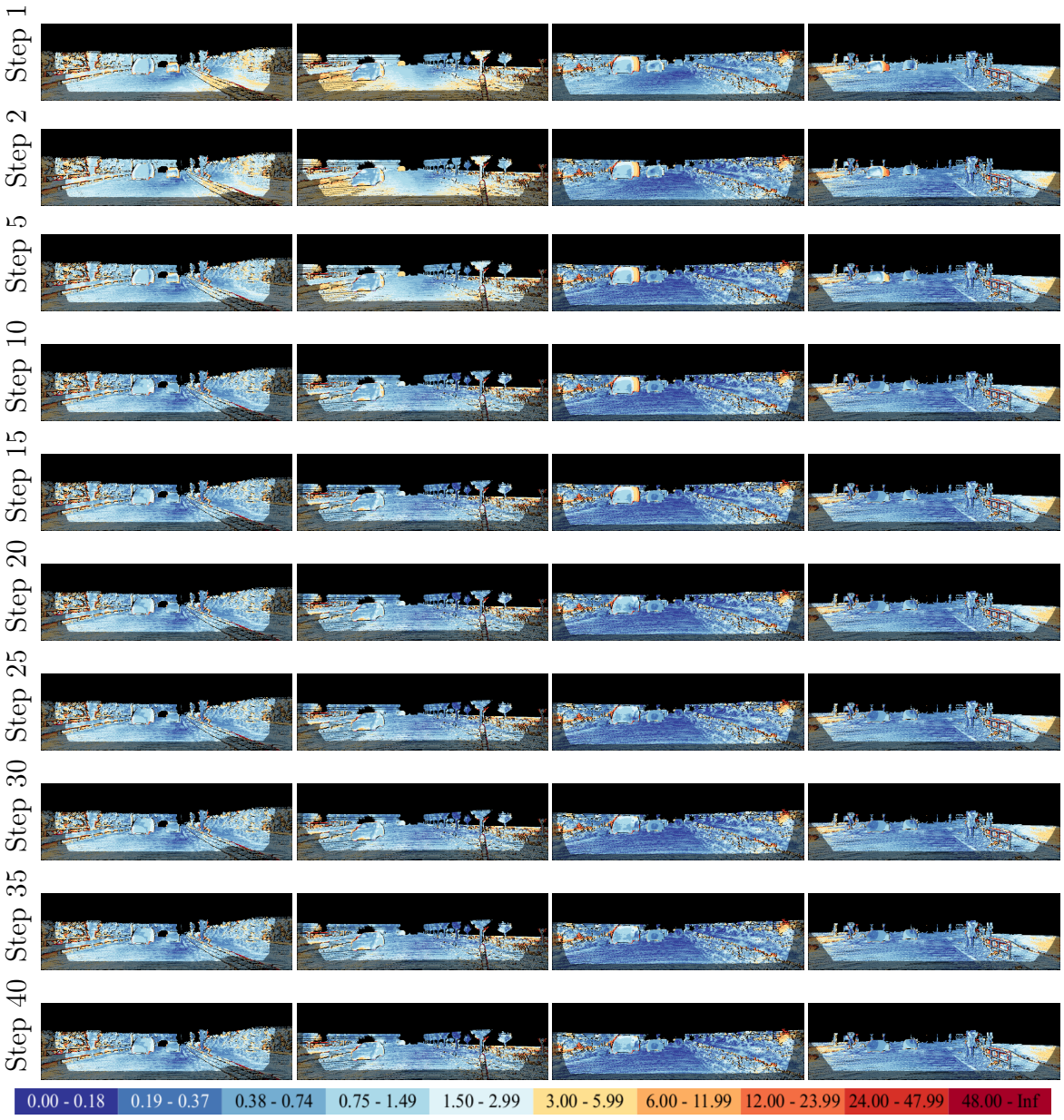


Figure B-3: **Qualitative scene flow error map at different GN iterations.** At first the estimated foreground motion is not accurate (see the orange/red cars in the first row). With our carefully designed energy terms, we are able to handle the outliers in the inferred visual cues and recover the accurate motion (see how the cars gradually turn into blue).

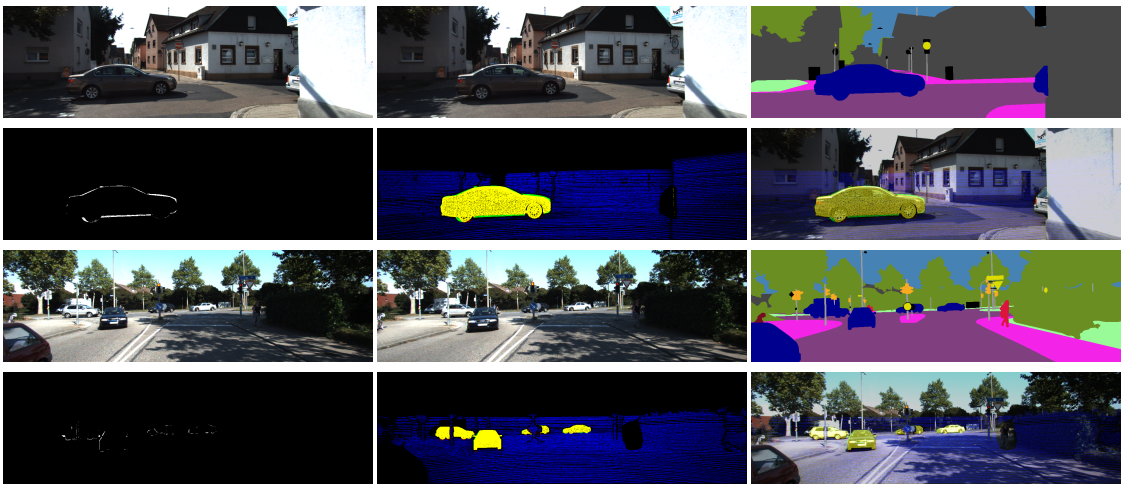


Figure B-4: **Mis-alignment between the GT scene flow instance and the GT segmentation.** We show two examples of the mis-alignment. A point is colored in yellow if both scene flow instance and GT segmentation agree it is foreground. Red means only segmentation agrees, and green suggests only scene flow instance agrees. A point is blue is both agree it is background. Points where the two disagree are shown in white in bottom left.

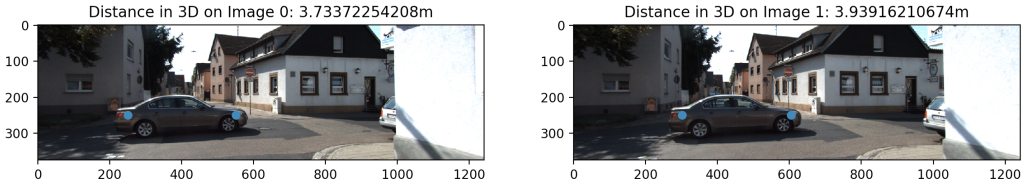


Figure B-5: **Evidence shows that some instances do not follow a rigid transform.** The 3D distance between exact same two points changes across frames. The distance increases by 5.4% within 0.1 second which is quite significant. Note that 3D position and correspondence are computed using GT disparity and GT flow.

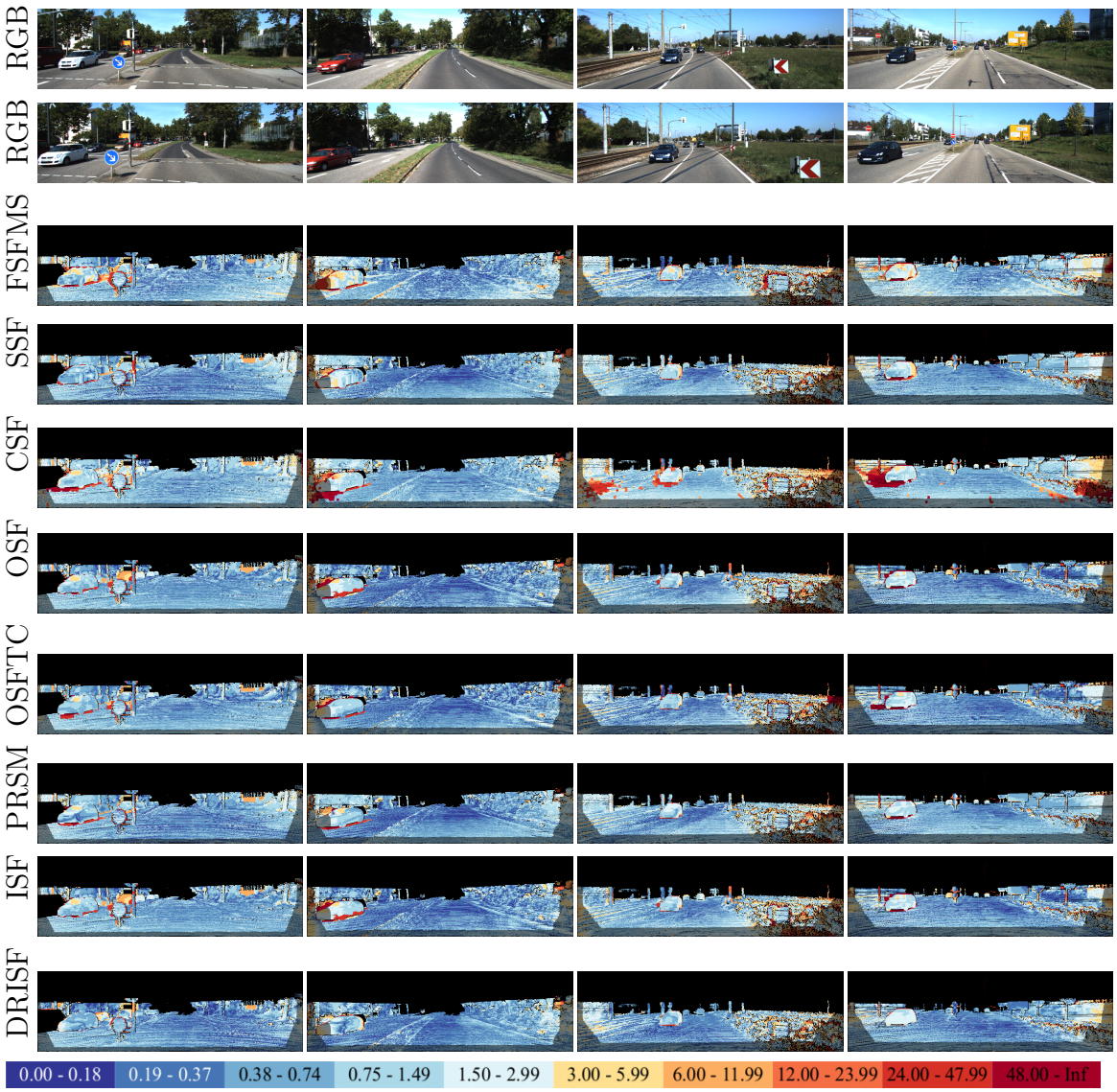


Figure B-6: **Qualitative comparison of scene flow on test set.** We compare with the competitive, top leading methods on KITTI leaderboard: FSFMS [342], SSF [295], CSF [228], OSF [243], OSFTC [256], PRSM [358], ISF [27]. Our method can effectively handle occlusion and texture-less regions. It is more robust to the illumination change as well as large displacement.

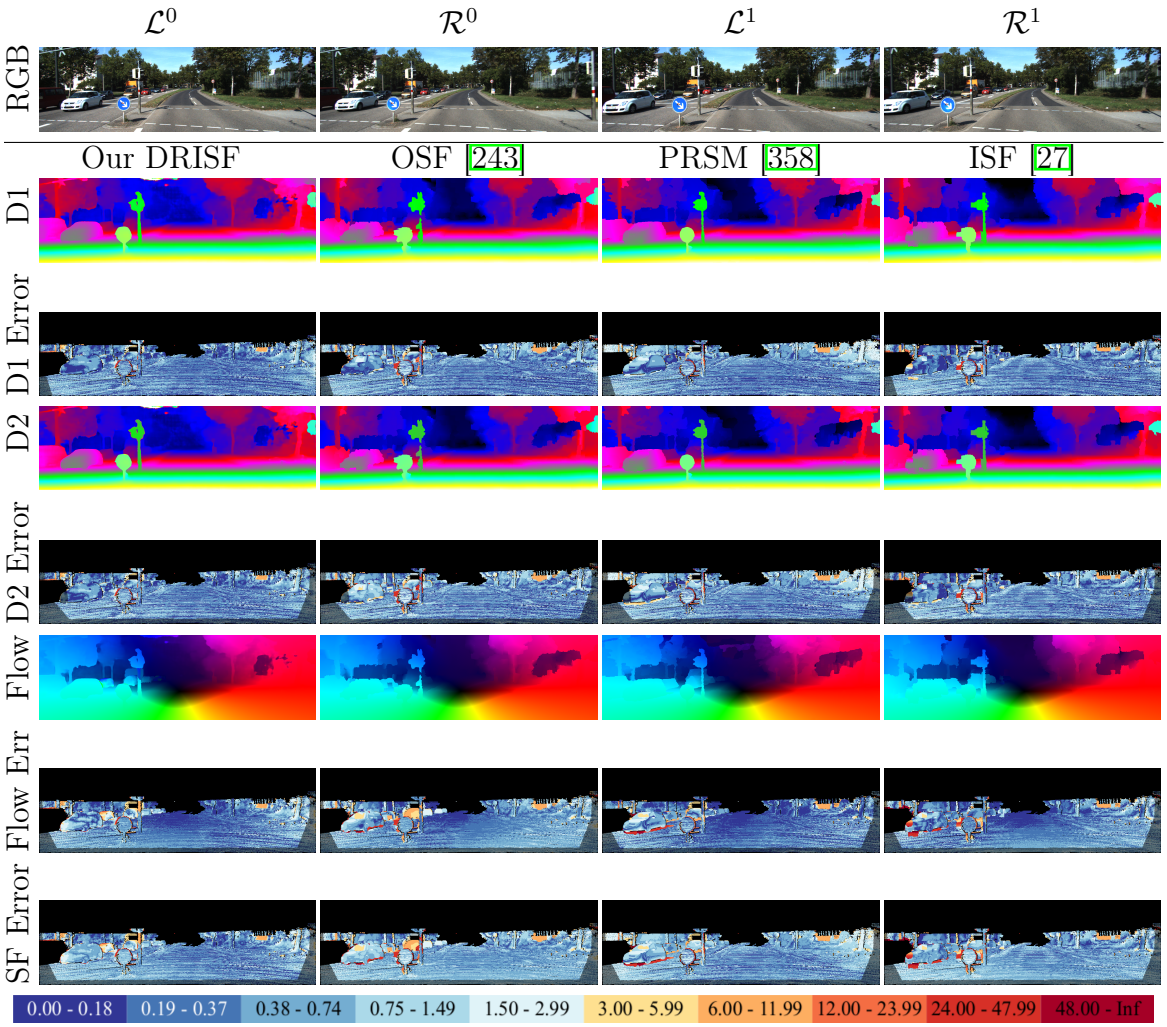


Figure B-7: **Qualitative comparison of depth and optical flow on test set.** We visualize the results and error maps of each component of OSF [243], PRSM [358], and ISF [27]. Our method can effectively handle occlusion and texture-less regions. It is more robust to the illumination change as well as large displacement.

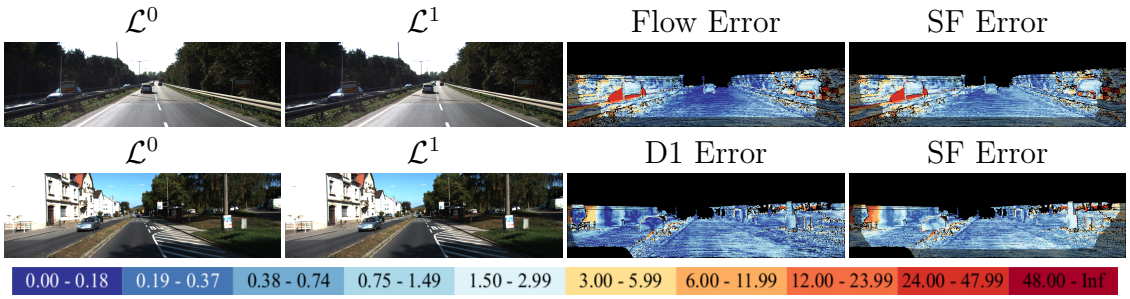
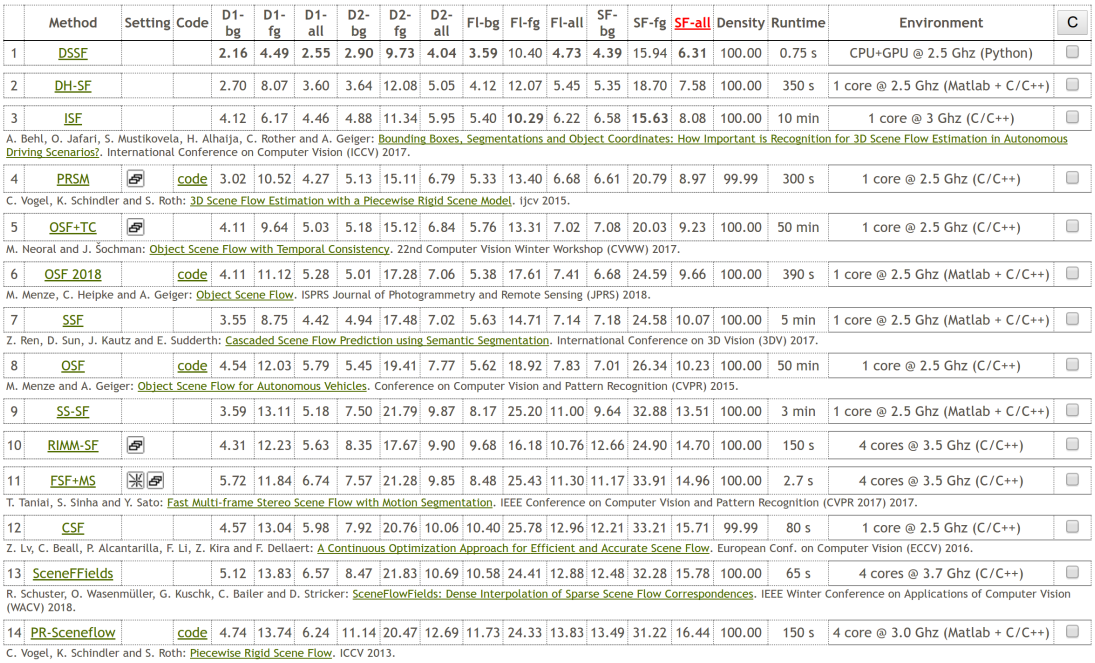


Figure B-8: **Failure cases of our motion estimation model.** Our energy model takes the inferred visual cues (*i.e.* flow, disparity) as input. If the original estimation is completely wrong, our model fails to recover the actual motion. In the first row, the flow estimation of the largely occluded vehicle is completely wrong. For background, the flow error has less impact, as we can recover its motion from other reliable background points. It however still relies heavily on D1 (see Eq. 6 in the paper). If D1 estimation is incorrect, our model can hardly work. To address this issue and avoid being bounded by the visual cues, we plan to optimize both the disparity and flow in the solver. We leave this for future study.

Figure B-9: **Failure case of the inferred 3D rigid motion model.** The small blue car is partially occluded by the white van in the first frame, and is completely invisible in the second. The estimated visual cues are thus completely wrong (unlike background it cannot benefit from other observations). It leads to 5 m translation error and 17.5 degree angular error.



	Method	Setting	Code	D1-bg	D1-fg	D1-all	D2-bg	D2-fg	D2-all	Fl-bg	Fl-fg	Fl-all	SF-bg	SF-fg	SF-all	Density	Runtime	Environment	C
1	DSSF			**2.16**	**4.49**	**2.55**	**2.90**	**9.73**	**4.04**	**3.59**	10.40	**4.73**	**4.39**	15.94	**6.31**	100.00	0.75 s	CPU+GPU @ 2.5 Ghz (Python)	
2	DH-SF			2.70	8.07	3.60	3.64	12.08	5.05	4.12	12.07	5.45	5.35	18.70	7.58	100.00	350 s	1 core @ 2.5 Ghz (Matlab + C/C++)	
3	ISF			4.12	6.17	4.46	4.88	11.34	5.95	5.40	**10.29**	6.22	6.58	**15.63**	8.08	100.00	10 min	1 core @ 3 Ghz (C/C++)	
	A. Behl, O. Jafari, S. Mustikovela, H. Alhaija, C. Rother and A. Geiger: Bounding Boxes, Segmentations and Object Coordinates: How Important is Recognition for 3D Scene Flow Estimation in Autonomous Driving Scenarios?. International Conference on Computer Vision (ICCV) 2017.																		
4	PRSM		code	3.02	10.52	4.27	5.13	15.11	6.79	5.33	13.40	6.68	6.61	20.79	8.97	99.99	300 s	1 core @ 2.5 Ghz (C/C++)	
	C. Vogel, K. Schindler and S. Roth: 3D Scene Flow Estimation with a Piecewise Rigid Scene Model. ijcv 2015.																		
5	OSF+TC			4.11	9.64	5.03	5.18	15.12	6.84	5.76	13.31	7.02	7.08	20.03	9.23	100.00	50 min	1 core @ 2.5 Ghz (C/C++)	
	M. Neoral and J. Šochman: Object Scene Flow with Temporal Consistency. 22nd Computer Vision Winter Workshop (CVWW) 2017.																		
6	OSF 2018		code	4.11	11.12	5.28	5.01	17.28	7.06	5.38	17.61	7.41	6.68	24.59	9.66	100.00	390 s	1 core @ 2.5 Ghz (Matlab + C/C++)	
	M. Menze, C. Heipke and A. Geiger: Object Scene Flow. ISPRS Journal of Photogrammetry and Remote Sensing (JPRS) 2018.																		
7	SSF			3.55	8.75	4.42	4.94	17.48	7.02	5.63	14.71	7.14	7.18	24.58	10.07	100.00	5 min	1 core @ 2.5 Ghz (Matlab + C/C++)	
	Z. Ren, D. Sun, J. Kautz and E. Sudderth: Cascaded Scene Flow Prediction using Semantic Segmentation. International Conference on 3D Vision (3DV) 2017.																		
8	OSF		code	4.54	12.03	5.79	5.45	19.41	7.77	5.62	18.92	7.83	7.01	26.34	10.23	100.00	50 min	1 core @ 2.5 Ghz (C/C++)	
	M. Menze and A. Geiger: Object Scene Flow for Autonomous Vehicles. Conference on Computer Vision and Pattern Recognition (CVPR) 2015.																		
9	SS-SF			3.59	13.11	5.18	7.50	21.79	9.87	8.17	25.20	11.00	9.64	32.88	13.51	100.00	3 min	1 core @ 2.5 Ghz (Matlab + C/C++)	
10	RIMM-SF			4.31	12.23	5.63	8.35	17.67	9.90	9.68	16.18	10.76	12.66	24.90	14.70	100.00	150 s	4 cores @ 3.5 Ghz (C/C++)	
11	FSF+MS			5.72	11.84	6.74	7.57	21.28	9.85	8.48	25.43	11.30	11.17	33.91	14.96	100.00	2.7 s	4 cores @ 3.5 Ghz (C/C++)	
	T. Taniai, S. Sinha and Y. Sato: Fast Multi-frame Stereo Scene Flow with Motion Segmentation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017) 2017.																		
12	CSF			4.57	13.04	5.98	7.92	20.76	10.06	10.40	25.78	12.96	12.21	33.21	15.71	99.99	80 s	1 core @ 2.5 Ghz (C/C++)	
	Z. Lv, C. Beall, P. Alcantarilla, F. Li, Z. Kira and F. Dellaert: A Continuous Optimization Approach for Efficient and Accurate Scene Flow. European Conf. on Computer Vision (ECCV) 2016.																		
13	SceneFFields			5.12	13.83	6.57	8.47	21.83	10.69	10.58	24.41	12.88	12.48	32.28	15.78	100.00	65 s	4 cores @ 3.7 Ghz (C/C++)	
	R. Schuster, O. Wasenmüller, G. Kuschk, C. Bailer and D. Stricker: SceneFlowFields: Dense Interpolation of Sparse Scene Flow Correspondences. IEEE Winter Conference on Applications of Computer Vision (WACV) 2018.																		
14	PR-Sceneflow		code	4.74	13.74	6.24	11.14	20.47	12.69	11.73	24.33	13.83	13.49	31.22	16.44	100.00	150 s	4 core @ 3.0 Ghz (Matlab + C/C++)	
	C. Vogel, K. Schindler and S. Roth: Piecewise Rigid Scene Flow. ICCV 2013.																		

Figure B-10: **Screenshot of the KITTI scene flow leaderboard at the time of paper submission.** Our model (named DSSF previously) achieves state-of-the-art performance on almost every entry (**bold**) while being significantly faster.

Appendix C

Supplementary: Deep Optimizer

C.1 Analyses

Deep feedback network as initialization: Structure optimization approaches are usually sensitive to initialization. The initial estimations have to be reasonably close so that the optimizers can converge to reasonably good results. Since our deep feedback network can bypass the local energy landscape and can produce very accurate estimations within extremely short amount of time, one natural solution is to exploit our model as an initialization and employ classic solvers for the final optimization. As shown in Tab. C.1, by combining with SoftRas [220], we can reduce the error by a significant margin. To further understand how robust the joint model is and how often it can converge to a certain error range, we visualize the cumulative error in Fig. C-1. The joint model significantly outperforms all approaches at 90th, 95th, and even 99th percentile, verifying our hypothesis that our deep feedback network is more robust to the initialization as well as the curvature of the local energy landscape, and can serve as a good initializer.

Effectiveness of unrolling steps: Tab. C.2 shows the step by step performance of our model on three different tasks. Through iterative feedback and update, our deep feedback inverse problem solver can significantly reduce the overall error.

Methods	Trans. Error Mean	Trans. Error Median	Rot. Error Mean	Rot. Error Median	Outlier (%)
NMR [168]	0.1	0.05	5.78	1.68	20.3
SoftRas [220]	0.05	0.003	4.14	0.5	8.03
Deep Regression	0.07	0.06	10.07	7.68	5
Ours	0.02	0.009	2.64	1.02	2.6
Ours + SoftRas	**0.01**	**0.001**	**1.62**	**0.31**	**0.9**

Table C.1: **Deep feedback inverse problem solver as an initializer.**

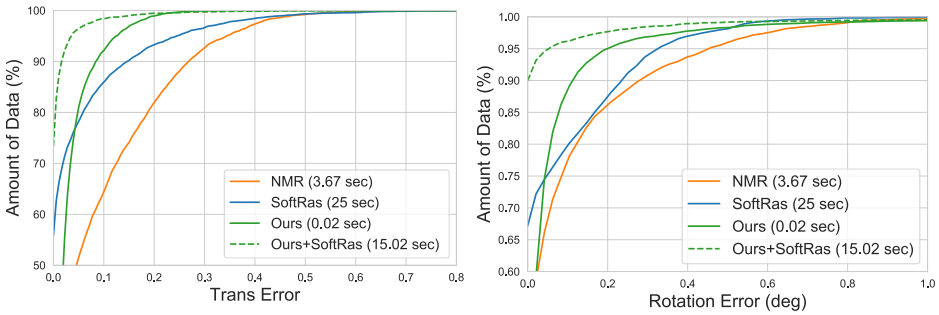


Figure C-1: **Cumulative error for translation and rotation.**

Shared weight vs non-shared weight: In the original paper, we argue that a non-shared weight network can model the dynamic output scale much easily. To corroborate our conjecture, we train a model following the exact same setting except the weight-sharing scheme. As shown in Tab. C.3, the non-shared weight network has a larger capacity and can better capture the dynamic distribution.

Category-wise performance: As shown in Tab. C.4, our model achieves similar performance across all categories except `bed` and `cabinet`. The two categories are extremely challenging because of the large intra-class variance. For instance, while `hammocks`, `loft bed`, and `ordinary bed` all belong to `bed` category, their shape and

6 DoF Pose Est.	Unroll Steps 1	3	5
Trans. error	0.075	0.032	0.02
Rot. error	10.07	3.32	2.64

Illum. Est.	Unroll Steps 1	3	7
Dir. light	0.076	0.055	0.052
Point light	0.111	0.089	0.084

Inv. kinematics	Unroll Steps 1	2	3
Position error	2.76	1.11	0.64
Rotation error	2.04	0.94	0.88

Table C.2: **Effectiveness of unrolling steps.**

Tasks	6 DoF Pose Est.		Illumination Est.		Inv. Kinematics	
Our model	Translation	Rotation	Directional	Point	Position	Rotation
Shared weight	0.03	4.24	0.077	0.103	0.99	1.57
Non-shared weight	**0.02**	**2.64**	**0.052**	**0.085**	**0.64**	**0.88**

Table C.3: **Effectiveness of not sharing weight across stages.**

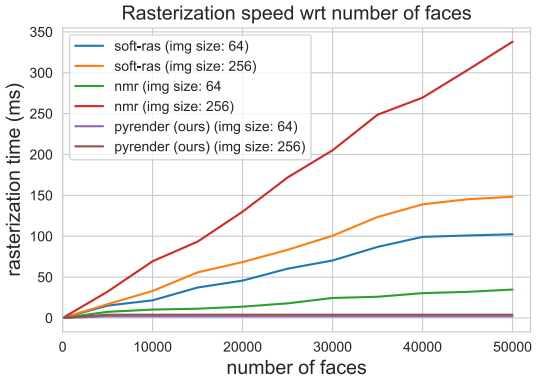


Figure C-2: **Runtime w.r.t. number of faces and image size.**

appearance vary significantly. In contrast, the shape variations among all cars are smaller and thus our model can pick up such shape prior much easily.

Influence of image size: Ideally we want to provide our model with as much information as possible. One straightforward solution is to exploit a larger rendered image. Unlike state-of-the-art differentiable renderers [168, 220] whose runtime increases significantly with image sizes, the computational overhead of pyrender is almost negligible (see Fig. C-2). This allows us to freely select the desired image size without sacrificing the speed. Tab. C.4 shows the results of our model with different input image sizes. The performance improves across all categories when the input image size increases from 64 to 128. Yet the results remains roughly the same when increasing the size from 128 to 256. Considering the slightly higher memory cost and slower speed, we select 128 as our final rendered image size.

	Optimization	Couch		Car		Bench		Monitor		Chair	
Image Size	Runtime	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.
64	16 ms	0.02	1.48	0.02	1.33	0.02	1.98	0.02	2.85	0.02	1.80
128	21 ms	**0.009**	**0.81**	**0.008**	**0.60**	**0.009**	1.10	**0.01**	1.49	**0.01**	0.95
256	35 ms	0.01	1.22	0.009	0.67	0.01	**1.05**	**0.01**	**1.33**	**0.01**	**0.92**
	Memory	Table		Sofa		Plane		Bed		Cabinet	
Image Size	Usage	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.
64	583 MB	0.02	2.76	0.02	1.61	0.08	3.83	0.08	7.78	0.05	5.83
128	636 MB	**0.01**	1.28	**0.009**	**0.85**	0.05	1.98	**0.07**	**4.56**	**0.02**	3.69
256	712 MB	**0.01**	**1.18**	0.01	0.87	**0.02**	**1.14**	0.08	4.88	**0.02**	**3.49**

Table C.4: **Effectiveness of image size on 6 DoF pose estimation.**

C.2 Related work

Connection to reinforcement learning (RL): Our method shares similarities with RL. Indeed, both frameworks are closed-loop systems, where the model takes the feedback from the "environment" and adjusts the next estimation accordingly. There exists, however, several key differences: First, the training strategy is different. While our approach exploits GT $\mathbf{x}_{\text{gt}}$ to directly train the feedback network $g_{\mathbf{w}}$ in a supervised fashion, RL agents learn by interacting with the closed-loop environment. The training signals of RL come from non-differentiable rewards $\mathbf{y} - \mathbf{y}^t$. Second, our model is trained to aggressively move towards the ground truth at each step. Thus we can accelerate the update procedure and reach the target with much fewer iterations. In comparison, RL agents are usually restricted to relatively small action space, due to the sampling efficiency and exploration issue, and typically require many more steps to arrive at the final solution. While one can augment the action space, it may bring difficulties to train RL agents in a sample efficient manner.

Comparison with DeepIM [194]: In this paper, we present a *generic framework* that is applicable to a wide range of inverse problems. While the instantiation of our model on 6D pose estimation is similar to the method Li *et al*. introduced in [194], there are a few key differences: (1) Li *et al*. *implicitly* model the relationships between the estimation and the observation. In contrast, we *explicitly* consider the difference and predict an update based upon it. Empirically we find that the explicit representation is crucial for learning and can drastically reduce the size of the model.

(2) Our model infers the 6d pose merely from silhouette images, yet [194] focuses on RGB images. (3) Our model is motivated by classic optimization approaches. We borrow ideas from traditional literature to improve the performance (*i.e.* adaptive update), whereas [194] simply unroll the network.

Comparison with IEF [47]: Our approach is related to [47]. Both work leverage feedback signals to refine the estimation iteratively. However, instead of relying on the network to implicitly establish the relationships between the feedback signal and the observation, we explicitly leverage the forward process to map the estimation back to the observation space and compare the difference. We empirically find that ensuring the inputs to lie in the same space is crucial for learning and can drastically reduce the size of the model. Moreover, unlike [47], our predicted update is not bounded. At each iteration, we aggressively move towards the GT solution and hence we can converge within a few iterations. While [47] argues that unbounded update is difficult, we find it possible thanks to the rich information encoded in the difference image. Finally, we focus on a wide range of inverse problems, whereas [47] is specialized to human pose estimation.

C.3 Other applications: JPEG image deblocking

Our approach is not limited to the tasks shown in the paper. It is designed in a generic fashion such that it can be applied to various inverse problems so long as the corresponding forward process f is given. This includes inverse image reconstruction problems such as inverse halftoning, JPEG compression noise removal, super resolution, etc. The underlying training and inference procedures for these tasks are the same as in the paper with the latent variable $\mathbf{x}$ now being high-dimensional (i.e., images).

To verify our claim, we test our approach on a classic inverse image processing task - JPEG image deblocking. Let $\mathbf{x}$ be a clean image, and $\mathbf{y} = f(\mathbf{x})$ be the compressed JPEG image generated by the forward non-differentiable JPEG compressor f. Our goal is to learn an inverse network g that can restore the original image $\mathbf{x} = g(\mathbf{y})$ from

the low-quality observation.

We follow a similar experimental setup to [422], where we train our model on BSD400 and evaluate it on BSD68 as well as Live29. We compare our method against state-of-the-art DnCNN [422] as well as Deep Image Prior (DIP) [352]. Our approach improves DIP by 2.33/4.15 db in PSNR and by 0.1076/0.1851 in SSIM (BSD68/Live29). Compared to DnCNN, we improve the PSNR by 0.04/0.03 db and SSIM by 0.001/0.001. We note that the above results are obtained by simple plug-and-play, without hyper-parameter tuning. We unroll our model 3 times.

C.4 Qualitative results

We provide more qualitative results of our model in Fig. C-3 and Fig. C-4. Through iterative feedback and update, our method is able to accurately infer the hidden parameters of interest. Furthermore, it can even recover from incorrect predictions in early stages. For instance, the initial pose estimations of the gray car and the air plane (last row in Fig. C-3) are completely wrong. Yet with the feedback signal, the model is able to iteratively refine the estimation and finally produce decent results.

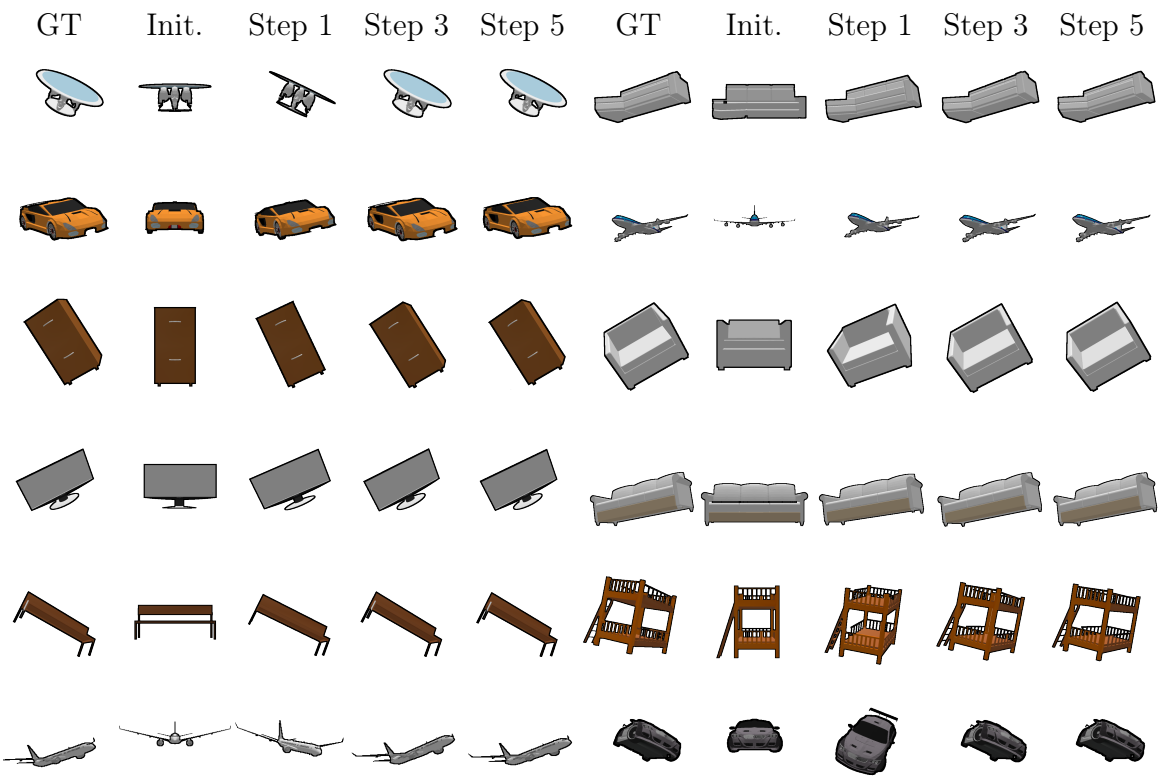


Figure C-3: **Qualitative results of 6 DoF pose estimation:** Our model can accurately estimate the 6 DoF pose of the object from a variety of viewpoints. It can also recover from incorrect estimation through iterative feedback. The images are rendered with the estimated/gt pose, purely for visualization purpose. We only exploit silhouette both during training and inference.

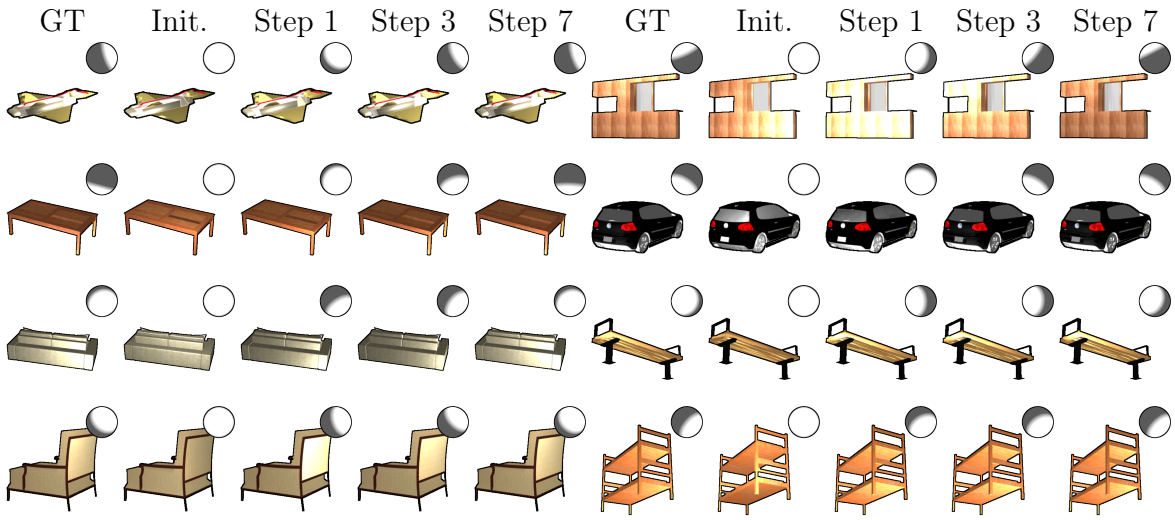


Figure C-4: **Qualitative results of illumination estimation.** While the initial estimations are not that accurate, our model is able to aggressively refine the prediction based on the feedback signal and achieve decent results. The input lights are visualized by rendering them onto a sphere (top-right).

Appendix D

Supplementary: Virtual Correspondences

D.1 Virtual Correspondences and Epipolar Geometry

In this section, we first briefly review epipolar geometry and then discuss how virtual correspondences relate to it.

Virtual Correspondence (VCs): VCs refer to a pair of pixels whose camera rays intersect in 3D. As mentioned in Sec. 3.1 (main paper), the intersection can happen at (i) co-visible 3D points, (ii) 3D points that are only visible in one image (and occluded in other other), or even (iii) invisible points (*e.g.*, free space, occupancy space, or points from occluded scene/objects). The first scenario is exactly the definition of classic correspondences. VCs, therefore, can be seen as a generalization of existing correspondences. We illustrate these scenarios in Fig. D-1.

Epipolar geometry: Epipolar geometry is the intrinsic projective geometry between two views [121]. It describes the relationship between existing correspondences, camera intrinsics, and the relative camera pose.

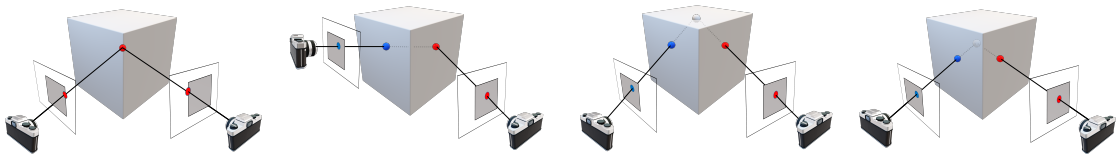


Figure D-1: **Illustration of different types of virtual correspondences.** VCs refer to a pair of pixels whose camera rays intersect in 3D. The intersection can happen at various places, which we illustrate a few here. The leftmost scenario is exactly the definition of classic correspondences. VCs, therefore, can be seen as a generalization of existing correspondences.

To be more specific, let $\boldsymbol{p}_1, \boldsymbol{p}_2 \in \mathbb{R}^2$ be a pair of correspondences in the two images. Based on epipolar geometry, the pixels can be related via

$$\bar{\boldsymbol{p}}_2^T \boldsymbol{F} \bar{\boldsymbol{p}}_1 = 0, \tag{D.1}$$

where $\bar{\cdot}$ represents the homogeneous coordinate, and $\boldsymbol{F} \in \mathbb{R}^{3\times3}$ is the fundamental matrix. Intuitively, $\boldsymbol{F}$ maps a point $\bar{\boldsymbol{p}}$ in one image to a line in the other $\boldsymbol{F}\bar{\boldsymbol{p}}$ (*i.e.*, the epipolar line); and $\bar{\boldsymbol{p}}_2^T \boldsymbol{F} \bar{\boldsymbol{p}}_1$ measures the distance between a pixel $\bar{\boldsymbol{p}}_2$ and an epipolar line $\boldsymbol{F}\bar{\boldsymbol{p}}_1$. If the two pixels are correspondences, they should lie on the epipolar line of each other and the point-line distance should be 0. We refer the readers to [121] for more details.

Virtual correspondences (VCs) meet epipolar geometry: VCs is a generalization of existing correspondences. Unlike traditional correspondences, VCs can describe different 3D surface points, and can have completely different appearances and semantics. Nevertheless, VCs still conform to epipolar geometry, which allows us to exploit classic geometric algorithms [224] to establish connections among them. Here, we provide a simple proof showing that VCs follow Eq. D.1, *i.e.*, the VC pixels lie on each other's epipolar line.

Proof. Given an image pair captured by camera 1 and camera 2, let $\mathbf{K}_1, \mathbf{K}_2 \in \mathbb{R}^{3\times3}$ be the camera intrinsics and $[\mathbf{R}_1, \mathbf{t}_1], [\mathbf{R}_2, \mathbf{t}_2] \in \mathbb{R}^{3\times4}$ be their respective extrinsic matrices. Without loss of generality, we assume the coordinate frame of camera 1 is

the world coordinate frame, thus $[\mathbf{R}_1, \mathbf{t}_1] = [\mathbf{I}, \mathbf{0}]$. Let $\boldsymbol{p}_1, \boldsymbol{p}_2 \in \mathbb{R}^2$ be the VCs and $\mathbf{X}_1^{\text{cam1}}, \mathbf{X}_2^{\text{cam2}} \in \mathbb{R}^3$ be their corresponding 3D points in the respective camera frames. The superscript denotes the coordinate frame the variable is in. Denote $\mathbf{o}_1^{\text{w}}, \mathbf{o}_2^{\text{w}} \in \mathbb{R}^3$ as the camera centers in the world coordinate frame, with $\mathbf{o}_1^{\text{w}} = \mathbf{0}$ and $\mathbf{o}_2^{\text{w}} = -\mathbf{R}_2^T\mathbf{t}_2$. By definition, we know that the camera rays of a pair of VCs intersect in 3D. Suppose the two camera rays intersect at $\mathbf{X}' = d_1\mathbf{X}_1^{\text{cam1}}$ in the world frame, we can write:

$$d_2\mathbf{X}_2^{\text{cam2}} = \mathbf{R}_2(d_1\mathbf{X}_1^{\text{cam1}}) + \mathbf{t}_2, \tag{D.2}$$

where d_1, d_2 are non-zero scalars.

The normal of the plane formed by the two camera rays can be computed as the cross product between the vector $\overrightarrow{\mathbf{o}_1^{\text{w}}\mathbf{o}_2^{\text{w}}}$ and the vector $\overrightarrow{\mathbf{o}_1^{\text{w}}\mathbf{X}'}$:

$$\begin{aligned}\overrightarrow{\mathbf{o}_1^{\text{w}}\mathbf{o}_2^{\text{w}}} \times \overrightarrow{\mathbf{o}_1^{\text{w}}\mathbf{X}'} &= (-\mathbf{R}_2^T\mathbf{t}_2) \times (d_1\mathbf{X}_1^{\text{cam1}}) \\ &= [-\mathbf{R}_2^T\mathbf{t}_2]_\times (d_1\mathbf{X}_1^{\text{cam1}}),\end{aligned} \tag{D.3}$$

where $[\cdot]_\times$ is the skew-symmetric matrix form defined as $[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$ for vector $\mathbf{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^T \in \mathbb{R}^3$.

Since the vector $\overrightarrow{\mathbf{o}_2^{\text{w}}\mathbf{X}'}$ also lies on the plane, the dot product between $\overrightarrow{\mathbf{o}_2^{\text{w}}\mathbf{X}'}$ and $\overrightarrow{\mathbf{o}_1^{\text{w}}\mathbf{o}_2^{\text{w}}} \times \overrightarrow{\mathbf{o}_1^{\text{w}}\mathbf{X}'}$ should be 0:

$$\begin{aligned}&\overrightarrow{\mathbf{o}_2^{\text{w}}\mathbf{X}'} \cdot (\overrightarrow{\mathbf{o}_1^{\text{w}}\mathbf{X}'} \times \overrightarrow{\mathbf{o}_1^{\text{w}}\mathbf{o}_2^{\text{w}}}) \\ &= \overrightarrow{\mathbf{o}_2^{\text{w}}\mathbf{X}'} \cdot ([-\mathbf{R}_2^T\mathbf{t}_2]_\times d_1\mathbf{X}_1^{\text{cam1}}) \\ &= (d_1\mathbf{X}_1^{\text{cam1}} + \mathbf{R}_2^T\mathbf{t}_2) \cdot ([-\mathbf{R}_2^T\mathbf{t}_2]_\times d_1\mathbf{X}_1^{\text{cam1}}) \\ &= 0.\end{aligned} \tag{D.4}$$

According to Eq. D.2, $d_1\mathbf{X}_1^{\text{cam1}} + \mathbf{R}_2^T\mathbf{t}_2 = \mathbf{R}_2^T d_2\mathbf{X}_2^{\text{cam2}}$. Eq. D.4 thus becomes:

$$\begin{aligned}
&\overrightarrow{\mathbf{o}_2^{\text{w}}\mathbf{X}'} \cdot (\overrightarrow{\mathbf{o}_1^{\text{w}}\mathbf{X}'} \times \overrightarrow{\mathbf{o}_1^{\text{w}}\mathbf{o}_2^{\text{w}}}) \\
&= (d_1\mathbf{X}_1^{\text{cam1}} + \mathbf{R}_2^T\mathbf{t}_2) \cdot ([-\mathbf{R}_2^T\mathbf{t}_2]_\times d_1\mathbf{X}_1^{\text{cam1}}) \\
&= (\mathbf{R}_2^T d_2\mathbf{X}_2^{\text{cam2}}) \cdot ([-\mathbf{R}_2^T\mathbf{t}_2]_\times d_1\mathbf{X}_1^{\text{cam1}}) \\
&= d_2\mathbf{X}_2^{\text{cam2},T}\mathbf{R}_2[-\mathbf{R}_2^T\mathbf{t}_2]_\times\mathbf{X}_1^{\text{cam1}}d_1 \\
&= 0.
\end{aligned} \tag{D.5}$$

Since d_1, d_2 are non-zero scalars, the above equation can be simplified as:

$$\mathbf{X}_2^{\text{cam2},T}\mathbf{R}_2[-\mathbf{R}_2^T\mathbf{t}_2]_\times\mathbf{X}_1^{\text{cam1}} = 0. \tag{D.6}$$

We can further re-write $\mathbf{X}_1^{\text{cam1}} = d_1'\mathbf{K}_1^{-1}\bar{\boldsymbol{p}}_1$ and $\mathbf{X}_2^{\text{cam2}} = d_2'\mathbf{K}_2^{-1}\bar{\boldsymbol{p}}_2$. Plugging these expressions into Eq. D.6 and removing the scalars d_1' and d_2', we arrive at:

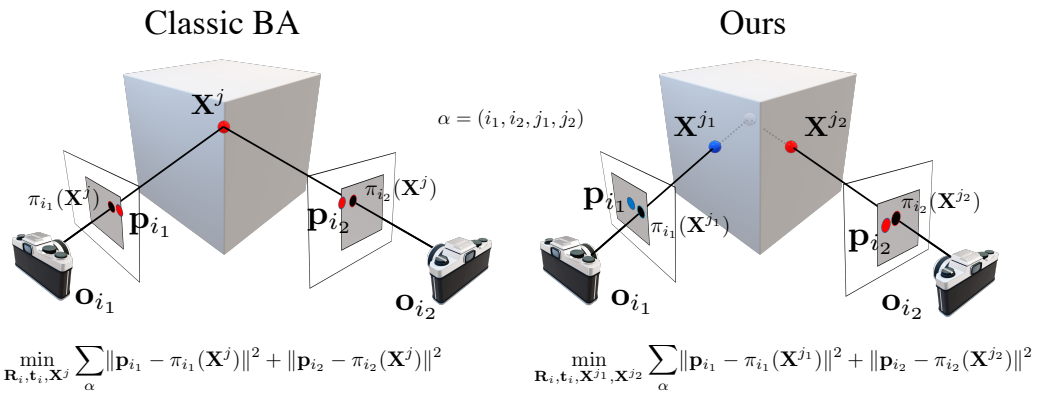
$$\begin{aligned}
\bar{\boldsymbol{p}}_2^T\mathbf{R}_2[-\mathbf{R}_2^T\mathbf{t}_2]_\times\bar{\boldsymbol{p}}_1 &= \bar{\boldsymbol{p}}_2^T\boldsymbol{F}\bar{\boldsymbol{p}}_1 \\
&= 0.
\end{aligned} \tag{D.7}$$

This is exactly the same equation as Eq. D.1. □

D.2 Bundle Adjustment for VCs

Classic bundle adjustment (BA): Classic bundle adjustment seeks to refine the camera poses and the *co-visible* 3D point locations through a joint least-square optimization procedure. In particular, it minimizes the distance between the reprojected points and the existing correspondences $\boldsymbol{p}_i$ for every image i in which the 3D point $\mathbf{X}^j$ is visible:

$$\min_{\mathbf{R}_i,\mathbf{t}_i,\mathbf{X}^j} \sum_\alpha \|\boldsymbol{p}_i - \pi_i(\mathbf{X}^j)\|^2. \tag{D.8}$$



(a) Classic BA optimizes the **co-visible** 3D point $\mathbf{X}^j$ and the camera poses $[\mathbf{R}_i\ \mathbf{t}_i]$ jointly

(b) Our BA optimizes the 3D **point tuple** $(\mathbf{X}^{j1}, \mathbf{X}^{j2})$ and the camera poses $[\mathbf{R}_i\ \mathbf{t}_i]$, subject to co-planar constraints.

Figure D-2: **Comparison to classic bundle adjustment.** See Sec. D.2 for more details.

Here, $\pi_i(\mathbf{X}^j) \sim \mathbf{K}_i(\mathbf{R}_i\mathbf{X} + \mathbf{t}_i)$ is the perspective projection operator, and $\alpha = (i, j)$ is the tuple of indices representing the (co-)visibility of each 3D point.

Efficient least-square solvers, such as Gauss-Newton and Levenberg–Marquardt algorithms are often exploited to tackle the BA algorithms. Minimizing this objective function also has a probabilistic interpretation: it is shown to be equivalent to the Maximum Likelihood Estimation (MLE) under Gaussian observation noise [3]. While BA is shown to be very powerful in practice, it heavily relies on classic correspondences to constrain the 3D points. If the input images barely overlap and only a few scene points are co-visible, BA may not be able to refine the estimations effectively.

BA for virtual correspondences: In this work, we combine virtual correspondences with bundle adjustment algorithms, significantly broadening the applicability of existing 3D systems. As we have derived in the main paper, the generalized BA objective is a constrained optimization problem:

$$\begin{aligned} &\min_{\mathbf{R}_i,\mathbf{t}_i,\mathbf{X}^{j_1},\mathbf{X}^{j_2}} \sum_\alpha \|\boldsymbol{p}_{i_1} - \pi_{i_1}(\mathbf{X}^{j_1})\|^2 + \|\boldsymbol{p}_{i_2} - \pi_{i_2}(\mathbf{X}^{j_2})\|^2 \\ &\text{s.t.} \quad \Big((\mathbf{X}^{j_1} - \mathbf{o}_{i_1}) \times (\mathbf{X}^{j_2} - \mathbf{o}_{i_2})\Big)^T (\mathbf{o}_{i_2} - \mathbf{o}_{i_1}) = 0, \end{aligned} \tag{D.9}$$

where $(\mathbf{X}^{j_1}, \mathbf{X}^{j_2})$ is the j-th pair of reconstructed 3D points, $(\boldsymbol{p}_{i_1}, \boldsymbol{p}_{i_2})$ is the associated VC pair from camera i_1 and camera i_2, and $\alpha = (i_1, i_2, j_1, j_2)$ is a tuple of corresponding indices. Formulating it as a constrained optimization problem, however, prohibits us from using classic gradient methods such as Gauss-Newton and Levenberg–Marquardt out of the box. Fortunately, since the two camera rays intersect in 3D and are co-planar, we can re-write one 3D point $\mathbf{X}^{j_2}$ as a function of the other $\mathbf{X}^{j_1}$ (see Eq. 3 of the main paper). Through this substitution, we arrive at:

$$\begin{aligned}
&\min_{\mathbf{R}_i, \mathbf{t}_i, \mathbf{X}^{j_1}, a^j, b^j} \sum_{\alpha} \|\boldsymbol{p}_{i_1} - \pi_{i_1}(\mathbf{X}^{j_1})\|^2 + \|\boldsymbol{p}_{i_2} - \pi_{i_2}(\mathbf{X}^{j_2})\|^2 \\
&= \min_{\mathbf{R}_i, \mathbf{t}_i, \mathbf{X}^{j_1}, a^j, b^j} \sum_{\alpha} \|\boldsymbol{p}_{i_1} - \pi_{i_1}(\mathbf{X}^{j_1})\|^2 \\
&+ \|\boldsymbol{p}_{i_2} - \pi_{i_2}(\mathbf{X}^{j_1} + a^j \cdot (\mathbf{X}^{j_1} - \mathbf{o}_{i_1}) + b^j \cdot (\mathbf{o}_{i_2} - \mathbf{o}_{i_1}))\|^2,
\end{aligned} \tag{D.10}$$

where a^j, b^j are free variables used to re-parameterize $\mathbf{X}^{j_2}$.

As we absorb the constraints into the objective, we can now exploit all gradient methods that have been used in classic BA. Furthermore, the new formulation (Eq. D.10) can handle both VCs and traditional correspondences and is a generalization of classic BA objective (Eq. D.8). To be more specific, if two pixels are classic correspondences, they describe the same, co-visible 3D point. In this case, $\mathbf{X}^{j_1} = \mathbf{X}^{j_2}$ and $a^j = b^j = 0$. Eq. D.10 reduces to Eq. D.8. An illustration of our model and classical BA is shown in Fig. D-2.

In practice, we treat the constraint as a soft constraint and optimize the objective below since we empirically find it works better:

$$\begin{aligned}
&\min_{\mathbf{R}_i, \mathbf{t}_i, \mathbf{X}^{j_1}, \mathbf{X}^{j_2}, a^j, b^j} \sum_{\alpha} \|\boldsymbol{p}_{i_1} - \pi_{i_1}(\mathbf{X}^{j_1})\|^2 + \|\boldsymbol{p}_{i_2} - \pi_{i_2}(\mathbf{X}^{j_2})\|^2 \\
&\qquad + \|\mathbf{X}^{j_2} - (a^j \cdot (\mathbf{X}^{j_1} - \mathbf{o}_{i_1}) + b^j \cdot (\mathbf{o}_{i_2} - \mathbf{o}_{i_1}))\|^2.
\end{aligned} \tag{D.11}$$

D.3 Quantitative Analyses

To better understand how our model and the baselines perform under different settings, we divide the data into various levels of difficulty based on the rotation difference of the GT cameras. Specifically, we categorize the CMU Panoptic dataset into three levels: easy ($0^\circ \sim 60^\circ$), medium ($60^\circ \sim 120^\circ$), and difficult ($120^\circ \sim 180^\circ$). As for the Mannequin Challenge dataset, since it is much smaller and most of the data falls into the easy category, we merge the medium and the difficult categories to balance the split. The two groups are: easy ($0^\circ \sim 60^\circ$), and difficult ($60^\circ \sim 180^\circ$). As shown in Tab. D.1 and Tab. D.2, our generalized SfM significantly outperforms the baselines when the viewpoint difference is large and is comparable to classic SfM in the traditional setup where the cameras are close and the viewpoint is alike. Our method performs slightly worse on the easy category of CMU dataset. We conjecture this is because CMU Panoptic dataset consists of many *accidental* viewpoints [101], which are very rare in real world, leading to noises in the 3D shape prediction model and the estimated VCs.

We also visualize the cumulative pose error in Fig. D-3. The cumulative plot of the baselines do not reach 100% because they failed to produce a reasonable estimate. For instance, if the number of traditional correspondences is less than 5 pairs, one cannot estimate the relative camera pose through essential matrix decomposition. Following previous work [308], we set the maximum translation error of the matching-based baselines to 90° since there is an inherent ambiguity when decomposing essential matrix. Compared with the baselines, our approach is way more robust. We achieve a 23.98° pose error at 80th percentile, while that of the classic SfM is over 120°.

Finally, we report the median error of our approach w.r.t. the number of input images in Fig. D-4. Both classic SfM and our generalized SfM benefit from more input images, since the scenes are more likely to overlap.

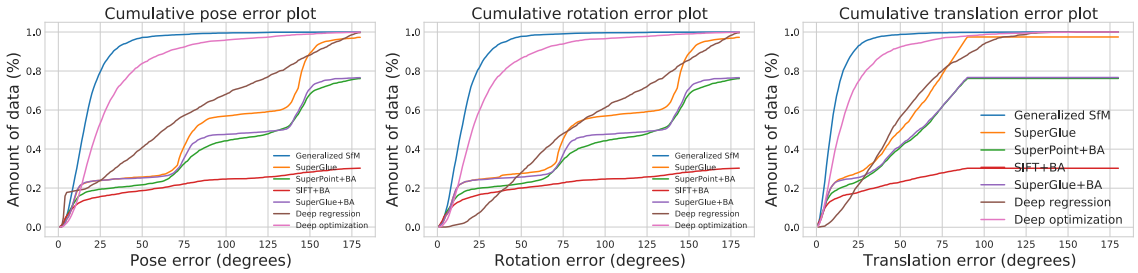


Figure D-3: **Cumulative pose error on CMU Panoptic dataset.**

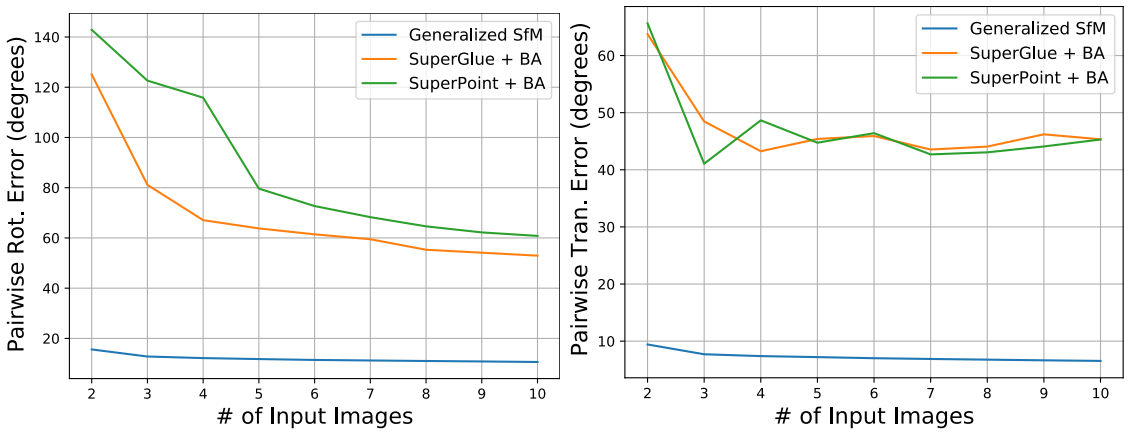


Figure D-4: **Performance vs number of images.**

D.4 Implementation Details

Optimizing the shape prior: As mentioned in Sec. 3.3 (main paper), the 3D points are obtained by marching rays from the camera centers through the pixels and recording the first intersections with the scene. Therefore, instead of directly optimizing the points as in Eq. D.10, an alternative is to optimize the relative low dimensional shape code and restrict the 3D points to lie within a certain range of the shape surface. The shape prior provides a strong regularization on the movements of the 3D points. However, we do not observe much difference in practice.

VC outlier filtering with coarse camera pose estimates: The initial VC estimates may contain noises due to imperfect initial 3D predictions. Our key observation is that the 3D pose estimation from monocular images, while not accurate enough, can still serve as a coarse pose prior. For instance, although deep nets may not be able to precisely estimate a person's 3D pose, it can still differentiate whether the

Methods	Easy (AUC)			Medium (AUC)			Difficult (AUC)			All (AUC)		
	@15°	@30°	@45°	@15°	@30°	@45°	@15°	@30°	@45°	@15°	@30°	@45°
SuperGlue [308]	34.91	56.22	63.97	3.10	6.69	8.47	0.00	0.00	0.05	10.02	16.74	19.36
SIFT [226] + BA [314]	29.00	42.36	48.61	0.70	1.41	2.14	0.04	0.19	0.38	7.68	11.39	13.33
SuperPoint [72] + BA [314]	34.48	50.88	57.81	1.17	2.07	2.79	0.00	0.09	0.23	9.22	13.77	15.85
SuperGlue [308] + BA [314]	**38.10**	**56.41**	63.02	2.66	6.07	7.94	0.00	0.00	0.00	10.68	16.57	18.92
Deep regression [179]	8.59	14.22	21.03	**14.47**	18.85	23.63	18.03	21.30	24.25	14.36	18.60	23.18
Deep optimization [30, 163]	7.21	28.22	44.27	7.44	25.94	41.81	8.83	27.70	41.86	7.88	27.17	42.42
Generalized SfM (ours)	25.51	52.29	**66.31**	13.13	**42.08**	**59.64**	**18.20**	**45.71**	**61.66**	**18.21**	**46.05**	**62.08**

Table D.1: **Pose estimation on CMU Panoptic dataset.**

Methods	Easy (AUC)			Difficult (AUC)			All (AUC)		
	@15°	@30°	@45°	@15°	@30°	@45°	@15°	@30°	@45°
SuperGlue [308]	**59.14**	73.90	80.37	4.14	8.14	10.94	26.38	34.85	39.10
SIFT [226] + BA [314]	27.69	33.19	35.65	0.52	0.67	0.72	14.17	20.24	24.25
SuperPoint [72] + BA [314]	34.58	47.71	55.04	0.0	1.5	3.2	17.12	23.48	26.81
SuperGlue [308] + BA [314]	58.61	72.87	78.94	4.15	9.25	12.41	26.24	35.12	39.46
Deep regression [179]	2.79	8.92	13.34	5.71	12.66	18.58	4.61	11.23	16.44
Deep optimization [30, 163]	29.08	61.81	74.54	11.02	42.19	60.03	15.38	47.08	63.67
Generalized SfM (ours)	53.57	**74.84**	**82.83**	**23.71**	**51.73**	**66.28**	**36.24**	**61.38**	**73.20**

Table D.2: **Pose estimation on Mannequin Challenge dataset.**

person is facing the camera or turning their back against it. We thus leverage these coarse pose predictions to prune outlier VCs that are extremely inconsistent with the pose predicted by the deep network. Specifically, we compute the symmetric epipolar distance for all the VCs using the camera pose predicted by the deep net and filter out the ones with extremely large error.

Filtering 3D predictions based on 2D visual cues: While one could exploit the 3D predictions from all images to construct VCs, it is sub-optimal in practice. The level of difficulty of single image 3D reconstruction varies significantly across different images and viewpoints. For instance, it is easier to reconstruct the human if we see the full body than only seeing parts; it is also less challenging if the person is not occluded. To this end, we present a simple yet effective strategy. We first construct VCs with the 3D shape of each image and estimate the corresponding camera motion. Then we project each estimated 3D object back to the cameras using the estimated pose and measure the consistency. If the consistency is low, it indicates that the estimated pose is wrong and therefore the VCs are unreliable.

Randomness: The only randomness within our approach is the RANSAC algorithm. For the deep networks, we simply use the pre-trained weights provided by the authors and conduct inference. We run the experiments 3 times and compute the standard deviation of the pose errors. The largest standard deviation is only 3% of the mean value. We conjecture this is because RANSAC already enumerates and compares various configurations. The numbers in the main paper are the median values. We omit the randomness error bar for simplicity, since many baselines do not have randomness.

Hyperparameters and GPU usage: We set the step size of L-BFGS to 0.001, and the number of iterations to 150. The hyperparameters are determined using the training/validation split of the CMU dataset*. We use 4 Titan V GPUs for experiments.

Extending VCs to other objects: As a proof-of-concept, we exploit canonical 3D deformable mapping (C3DM) [261] and adapt our method to cars. C3DM is a deep learning based non-rigid SfM model that allows one to infer the coarse shape and pose of a car from a single monocular image. It also provides a mapping between each pixel and the surface of the vehicle. By replacing our current SMPL and DensePose with C3DM, we can directly apply our system to cars without any bells and whistles. We show another example in Fig. D-5 where we acheive a pose error of 13 degrees, whereas the pose error of C3DM is 25 degrees. We conjecture the improvement is because we only use the shape and pose from C3DM for VCs estimation. Through further bundle adjustment, we are able to refine the poses and mitigate the noises.

Reproducibilty: We have provided comprehensive details in the main paper as well as the supp. material. To further facilitate reproducibility, we will release our code as well as the curated data upon publication.

*Our method does not have a training stage. We simply use the data for hyperparamter search.

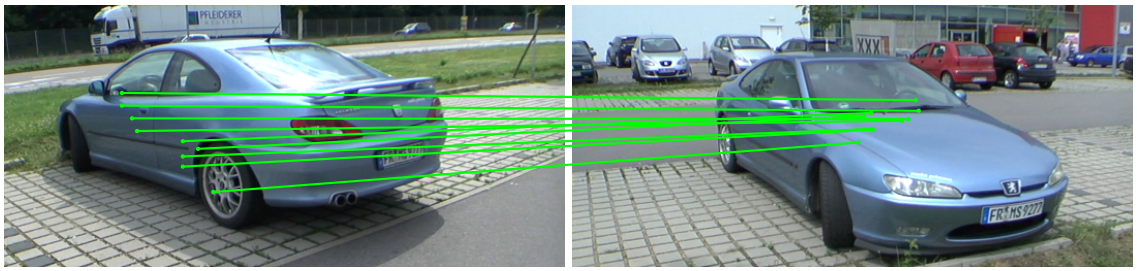


Figure D-5: **Virtual correspondences from cars.**

D.5 Dataset Statistics

We show a randomly sampled subset of images from our curated datasets in Fig. D-6 and Fig. D-7. While CMU Panoptic dataset consists of a wide range of viewpoints (even from the top), Mannequin Challenge dataset comprises much more diverse indoor and outdoor background. We also compute the statistics of the GT camera poses. As shown in Fig. D-8, the spatial distribution of the cameras between the two splits of CMU dataset are similar. We visualize them in Fig. D-8(left), where each color corresponds to a split. Comparing to the CMU Panoptic dataset, the translations between cameras are much smaller in the Mannequin Challenge dataset (see Fig. D-9) due to how the data is collected.

Dataset sources and licenses In this work we perform evaluation on data from several sources. For quantitative evaluation, we use the public CMU Panoptic dataset [155, 156] and follow its non-commercial research-only license. We additionally curate sequences from the public Mannequin Challenge dataset [199], which contains raw public youtube videos licensed by Google LLC under a Creative Commons Attribution 4.0 International License. Moreover, to obtain more real-world data, we filmed 6 additional Mannequin Challenge videos, and all human subjects appearing in the video have agreed to data collection and release for research purposes. For illustration purposes and qualitative evaluation, we have included several examples from movies and sports events, and we hereby cite them. Fig. 1 in the main paper contains images from (1) Kobe Bryant's last shot (images from Lakers Nation), and (2) the 1997 movie *Good Will Hunting* directed by Gus Van Sant and produced by

Figure D-6: **Snapshot of CMU Panoptic Dataset.**

Figure D-7: **Snapshot of Mannequin Challenge dataset.**

Lawrence Bender, starring Ben Affleck and Matt Damon. Fig. D-12 contains pictures from (1) the 1994-2004 *Friends* series produced by Bright/Kauffman/Crane Productions starring Jennifer Aniston and David Schwimmer; (2) Michael Jordan's last shot (image on the left from the google images; image on the right from *Sports Illustrated* photographer Walter Iooss Jr.), and (3) the 2013 movie *The Wolf of Wall Street* directed by Martin Scorsese, produced by Paramount Pictures and starring Leonardo DiCaprio and Matthew McConaughey. We have obtained all movie and sports images from the internet, and we will remove these results if copyright is infringed.

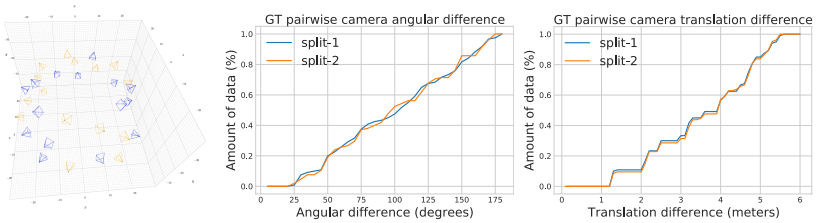


Figure D-8: **Statistics of CMU Panoptic Dataset.**

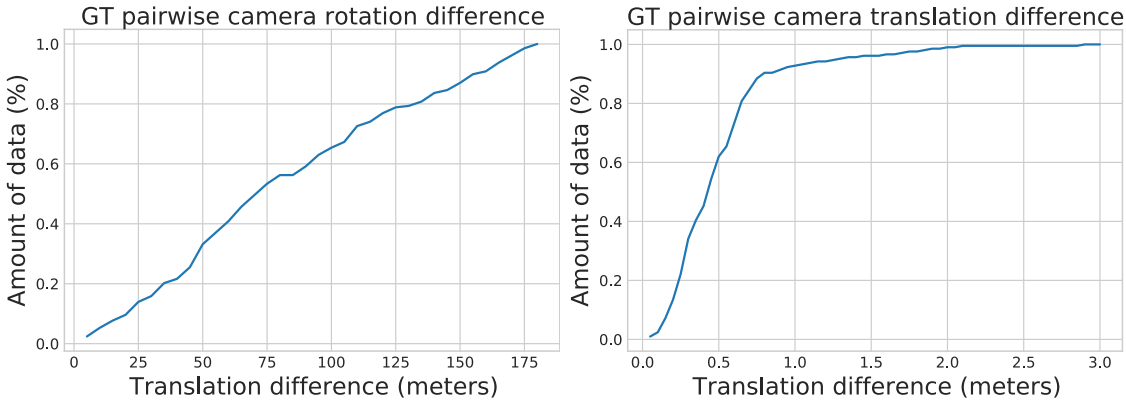


Figure D-9: **Statistics of Mannequin Challenge dataset.**

D.6 Qualitative Results

We show more qualitative results in Fig. D-10, Fig. D-11 and Fig. D-12.

D.7 Social Impact

Our proposed method reduces the need to capture dense views for camera pose estimation and 3D reconstruction. Hence it has the potential to reduce storage and computational costs. However, it could have negative societal impact. Similar to all 3D reconstruction applications, our method could be exploited by surveillance; it could also raise privacy concerns as 3D reconstruction from few images becomes more accessible.

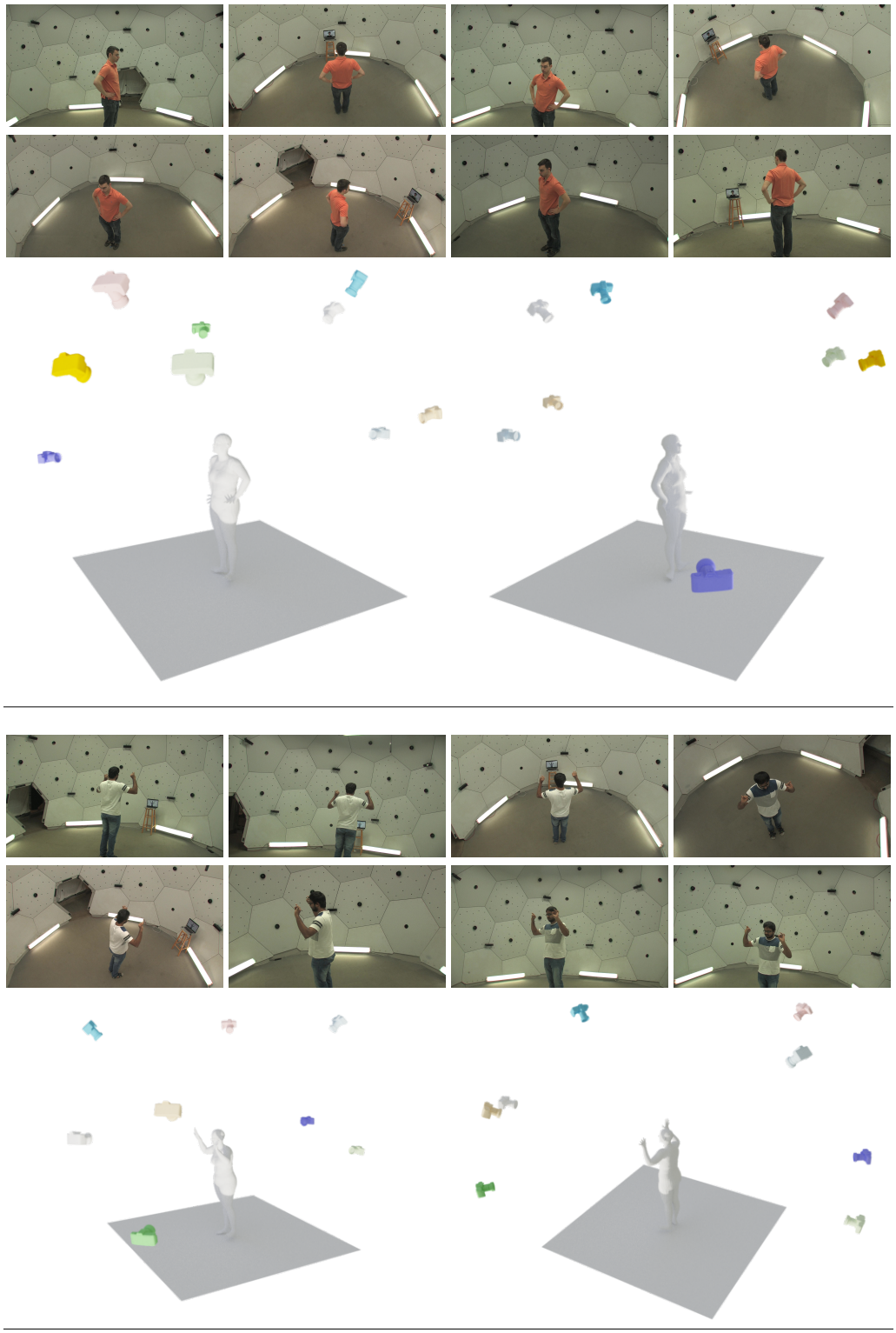


Figure D-10: **Qualitative Results on CMU Panoptic dataset.**

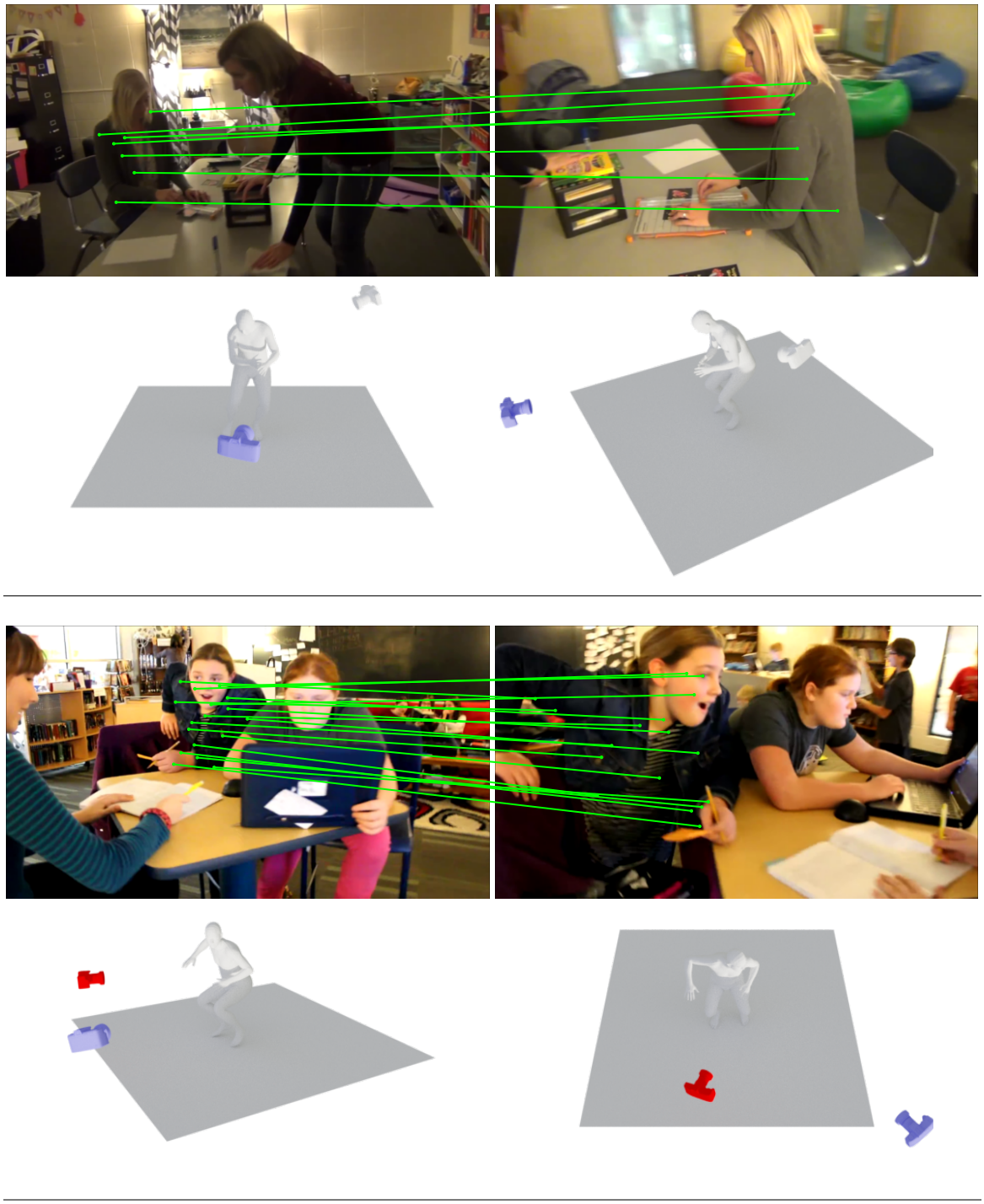


Figure D-11: **Qualitative Results on Mannequin Challenge dataset.**

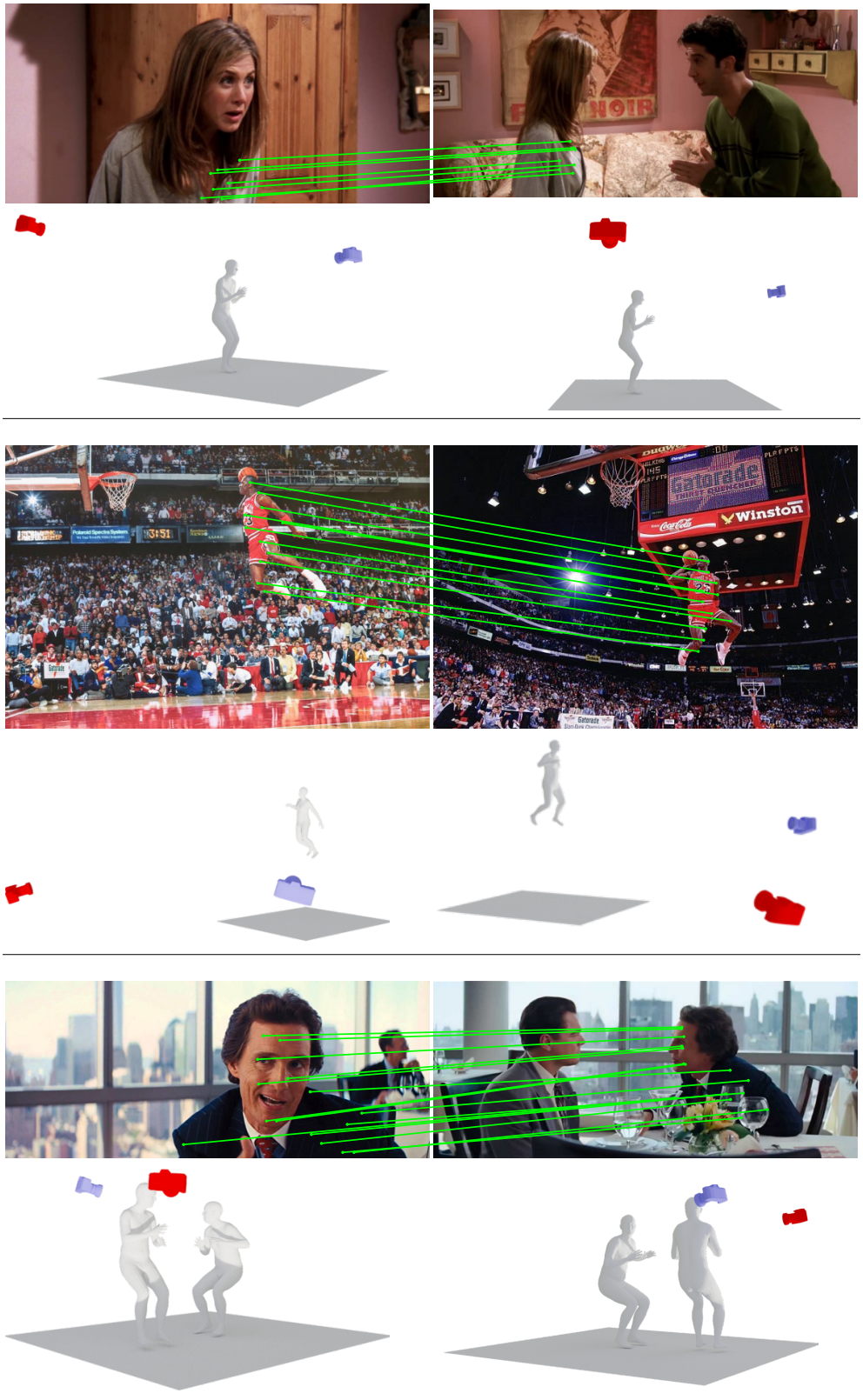


Figure D-12: **Qualitative Results on movies, sitcom, and sports photograph.**

D.8 Interactive Results

We also provide a project page that contains more qualitative and interactive visualizations. We strongly encourage the readers to take a look. Please visit the webiste that we host: https://virtual-correspondence.github.io.

Appendix E

Supplementary: Structure from Duplicates

E.1 Dataset

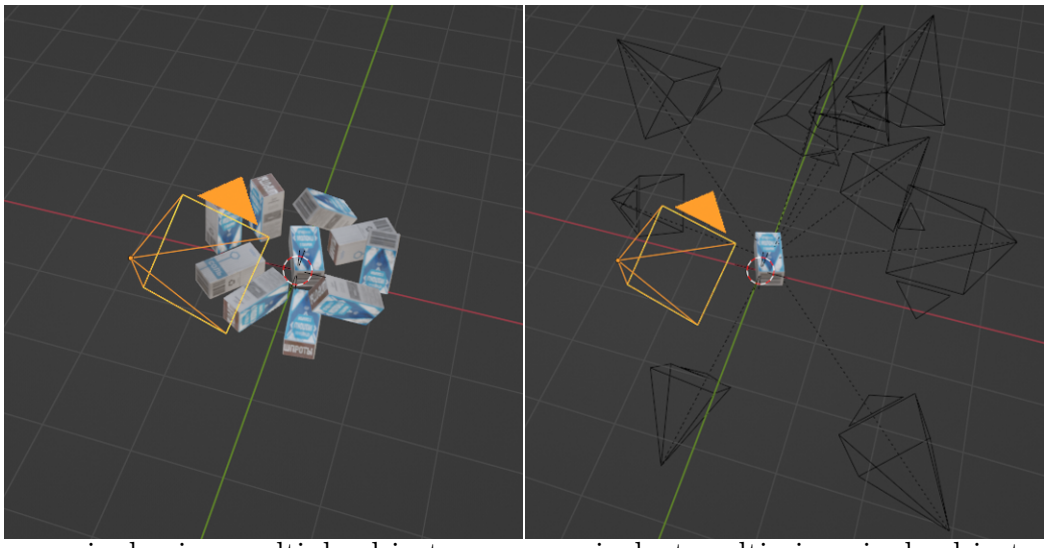
Our new dataset, **Dup**, comprises 13 synthetic scenes and 6 real-world scenes.

E.1.1 Synthetic data

The synthetic data includes `apple`, `medicine box`, `can`, `driller`, `color box`, `cash machine`, `cleaner`, `clock`, `coffee machine`, `wood guitar`, `warning sign`, `fire extinguisher`, and `food tin`. In particular, `apple`, `medicine box`, `can`, and `driller` each have 100 training views and 200 testing views, making them suitable for the traditional multi-view dense observation setup. The remaining scenes each consist of 7-10 multi-view images. We use them to explore the relationship between single-view multiple instances and multi-view single objects.

E.1.2 Real-world data

We randomly placed the objects on the table and use our mobile phone to capture the data. In total we collect six real-world scenes: `toy airplane`, `cake box`, `cheese box`, `cola`, `potato chips`, and `yogurt`. The number of instances in each scene ranges from



single-view multiple objects equivalent multi-view single object

Figure E-1: Without considering the lighting effect and occlusion, a single image with duplicated objects (left) is equivalent to multi-view observations of a single object (right).

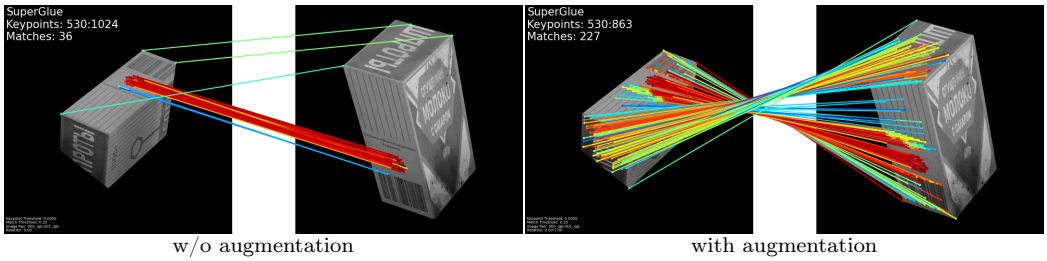
five to ten.

E.2 Analyses

On the duality between single-view multi-instances and multi-view single instance: As depicted in Fig. E-1, when the lighting effect and occlusion are disregarded, a single image containing duplicated objects can be treated as observing a single object from multiple viewpoints.

Importance of rotation augmentation: Incorporating in-plane rotation augmentation significantly improves the correspondence matching process. Fig. E-2 shows some qualitative results.

Singe image inverse rendering: We show more qualitative results in Fig. E-3 and Fig. E-4.



w/o augmentation with augmentation

Figure E-2: **Importance of rotation augmentation for structure from motion:** After in-plane rotation augmentation, we are able to estimate more correspondences between two instances. Here, we adopt Superpoints [71] and SuperGlue [307] for matching.

Figure E-3: **Single-view inverse rendering of the "cleaner" scene.**

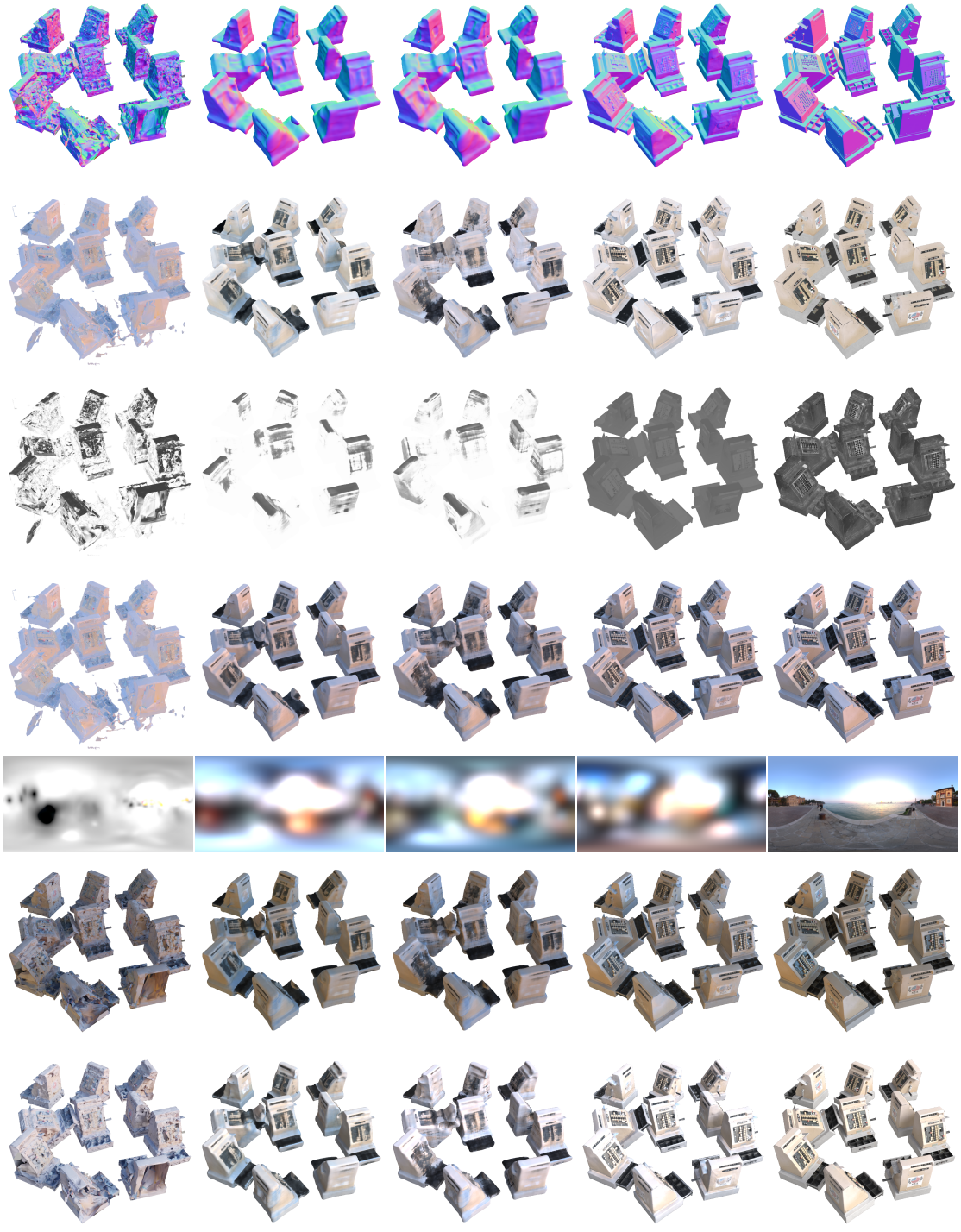


Figure E-4: **Single-view inverse rendering of the "cash machine" scene.**

Bibliography

[1] Pyrender. https://github.com/mmatl/pyrender, 2020.

[2] Explore the possibilities of realitycapture, 2021. URL https://www.capturingreality.com/.

[3] Sameer Agarwal, Noah Snavely, Steven M Seitz, and Richard Szeliski. Bundle adjustment in the large. In *ECCV*, 2010.

[4] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *ACM Communications*, 2011.

[5] Farhad Aghili and Chun-Yi Su. Robust relative navigation by integration of icp and adaptive kalman filter using laser scanner and imu. *IEEE/ASME Transactions on Mechatronics*, 2016.

[6] Ijaz Akhter, Yaser Sheikh, Sohaib Khan, and Takeo Kanade. Nonrigid structure from motion in trajectory space. In *NIPS*, 2008.

[7] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *ECCV*, 2020.

[8] Alexander Amini, Igor Gilitschenski, Jacob Phillips, Julia Moseyko, Rohan Banerjee, Sertac Karaman, and Daniela Rus. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *RA-L*, 2020.

[9] Alexander Amini, Tsun-Hsuan Wang, Igor Gilitschenski, Wilko Schwarting, Zhijian Liu, Song Han, Sertac Karaman, and Daniela Rus. VISTA 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In *ICRA*, 2022.

[10] Roland Angst and Marc Pollefeys. Static multi-camera factorization using rigid motion. In *ICCV*, 2009.

[11] Roland Angst and Marc Pollefeys. Multilinear factorizations for multi-camera rigid structure from motion problems. *IJCV*, 2013.

[12] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, 2016.

[13] Georges Baatz, Kevin Köser, David Chen, Radek Grzeszczuk, and Marc Pollefeys. Leveraging 3d city models for rotation invariant place-of-interest recognition. *IJCV*, 2012.

[14] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, 2017.

[15] Min Bai, Wenjie Luo, Kaustav Kundu, and Raquel Urtasun. Exploiting semantic information and deep matching for optical flow. In *ECCV*, 2016.

[16] Min Bai, Gellert Mattyus, Namdar Homayounfar, Shenlong Wang, Shrinidhi Kowshika Lakshmikanth, and Raquel Urtasun. Deep multi-sensor lane detection. In *IROS*, 2018.

[17] Mohamed El Banani, Jason J Corso, and David F Fouhey. Novel object viewpoint estimation through reconstruction alignment. In *CVPR*, 2020.

[18] Mayank Bansal and Kostas Daniilidis. Geometric urban geo-localization. In *CVPR*, 2014.

[19] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *TPAMI*, 2014.

[20] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021.

[21] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022.

[22] Harry Barrow, J Tenenbaum, A Hanson, and E Riseman. Recovering intrinsic scene characteristics. *Comput. vis. syst*, 1978.

[23] Ioan Andrei Barsan, Shenlong Wang, Andrei Pokrovsky, and Raquel Urtasun. Learning to localize using a lidar intensity map. In *CoRL*, 2018.

[24] Tali Basha, Yael Moses, and Nahum Kiryati. Multi-view scene flow estimation: A view centered variational approach. *IJCV*, 2013.

[25] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[26] Paul Beardsley, Phil Torr, and Andrew Zisserman. 3d model acquisition from extended image sequences. In *ECCV*, 1996.

[27] Aseem Behl, Omid Hosseini Jafari, Siva Karthik Mustikovela, Hassan Abu Alhaija, Carsten Rother, and Andreas Geiger. Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios? In *ICCV*, 2017.

[28] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *TOG*, 2014.

[29] James R Bergen and Edward H Adelson. Early vision and texture perception. *Nature*, 1988.

[30] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, 1992.

[31] Michael J Black and Paul Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *CVIU*, 1996.

[32] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH*, 1999.

[33] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[34] Eric Brachmann and Carsten Rother. Neural-guided ransac: Learning where to sample model hypotheses. In *ICCV*, 2019.

[35] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *CVPR*, 2000.

[36] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004.

[37] Marcus A Brubaker, Andreas Geiger, and Raquel Urtasun. Lost! leveraging the crowd for probabilistic visual self-localization. In *CVPR*, 2013.

[38] Brent Burley and Walt Disney Animation Studios. Physically-based shading at disney. In *Siggraph*, 2012.

[39] Wonmin Byeon, Thomas M Breuel, Federico Raue, and Marcus Liwicki. Scene labeling with lstm recurrent neural networks. In *CVPR*, 2015.

[40] Lucas Caccia, Herke Van Hoof, Aaron Courville, and Joelle Pineau. Deep generative modeling of lidar data. In *IROS*, 2019.

[41] Ruojin Cai, Bharath Hariharan, Noah Snavely, and Hadar Averbuch-Elor. Extreme rotation estimation using dense correlation volumes. In *CVPR*, 2021.

[42] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *ECCV*, 2010.

[43] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier analysis and applications*, 2008.

[44] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *TPAMI*, 2019.

[45] Zhe Cao, Yaser Sheikh, and Natasha Kholgade Banerjee. Real-time scalable 6dof pose estimation for textureless objects. In *ICRA*, 2016.

[46] Wayne E Carlson. An algorithm and data structure for 3d object synthesis using surface patch intersections. In *SIGGRAPH*, 1982.

[47] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *CVPR*, 2016.

[48] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *ECCV*, 2020.

[49] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, 2022.

[50] Tony F Chan, Jianhong Shen, and Hao-Min Zhou. Total variation wavelet inpainting. *Journal of Mathematical imaging and Vision*, 2006.

[51] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv*, 2015.

[52] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11315–11325, 2022.

[53] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, 2018.

[54] Kefan Chen, Noah Snavely, and Ameesh Makadia. Wide-baseline relative camera pose estimation with directional learning. In *CVPR*, 2021.

[55] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv*, 2023.

[56] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. *NeurIPS*, 2016.

[57] Wenzheng Chen, Joey Litalien, Jun Gao, Zian Wang, Clement Fuji Tsang, Sameh Khamis, Or Litany, and Sanja Fidler. Dib-r++: learning to predict lighting and material with a hybrid differentiable renderer. *NeurIPS*, 2021.

[58] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017.

[59] Yun Chen, Frieda Rong, Shivam Duggal, Shenlong Wang, Xinchen Yan, Sivabalan Manivasagam, Shangjie Xue, Ersin Yumer, and Raquel Urtasun. Geosim: Realistic video simulation via geometry-aware composition for self-driving. In *CVPR*, 2021.

[60] Zhuoyuan Chen, Xun Sun, Liang Wang, Yinan Yu, and Chang Huang. A deep visual correspondence embedding model for stereo matching costs. In *ICCV*, 2015.

[61] Bowen Cheng, Anwesa Choudhuri, Ishan Misra, Alexander Kirillov, Rohit Girdhar, and Alexander G Schwing. Mask2former for video instance segmentation. *arXiv*, 2021.

[62] Yen-Chi Cheng, Chieh Hubert Lin, Hsin-Ying Lee, Jian Ren, Sergey Tulyakov, and Ming-Hsuan Yang. In&out: Diverse image outpainting via gan inversion. *arXiv*, 2021.

[63] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3D shape reconstruction and completion. In *CVPR*, 2020.

[64] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. Universal correspondence network. 2016.

[65] MAR Cooper and S Robson. Theory of close range photogrammetry. 1996.

[66] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[67] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. *ICRA*, 2019.

[68] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *IJRR*, 2008.

[69] Alessio Del Bue. A factorization approach to structure from motion with shape priors. In *CVPR*, 2008.

[70] Alessio Del Bue, Xavier Llad, and Lourdes Agapito. Non-rigid metric shape and motion recovery from uncalibrated images using priors. In *CVPR*, 2006.

[71] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPRW*, 2018.

[72] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPR*, 2018.

[73] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014.

[74] Junting Dong, Qing Shuai, Yuanqing Zhang, Xian Liu, Xiaowei Zhou, and Hujun Bao. Motion capture from internet videos. In *ECCV*, 2020.

[75] Junting Dong, Qi Fang, Wen Jiang, Yurou Yang, Hujun Bao, and Xiaowei Zhou. Fast and robust multi-person 3d pose estimation and tracking from multiple views. In *TPAMI*, 2021.

[76] Weisheng Dong, Lei Zhang, Guangming Shi, and Xiaolin Wu. Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. *TIP*, 2011.

[77] Yue Dong, Guojun Chen, Pieter Peers, Jiawan Zhang, and Xin Tong. Appearance-from-motion: Recovering spatially varying surface reflectance under unknown lighting. *TOG*, 2014.

[78] David L Donoho. De-noising by soft-thresholding. *Transactions on Information Theory*, 1995.

[79] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[80] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv*, 2020.

[81] Shivam Duggal and Deepak Pathak. Topologically-aware deformation fields for single-view 3d reconstruction. *CVPR*, 2022.

[82] Shivam Duggal, Zihao Wang, Wei-Chiu Ma, Sivabalan Manivasagam, Justin Liang, Shenlong Wang, and Raquel Urtasun. Mending neural implicit modeling for 3d vehicle reconstruction in the wild. In *WACV*, 2022.

[83] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. *arXiv*, 2019.

[84] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001.

[85] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[86] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10786–10796, 2021.

[87] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *NeurIPS*, 2014.

[88] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *TIP*, 2006.

[89] Sovann En, Alexis Lechervy, and Frédéric Jurie. Rpnet: An end-to-end network for relative camera pose estimation. In *ECCVW*, 2018.

[90] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014.

[91] Dave Epstein, Boyuan Chen, and Carl Vondrick. Oops! predicting unintentional action in video. *arXiv*, 2019.

[92] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

[93] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

[94] Jin Fang, Dingfu Zhou, Feilong Yan, Tongtong Zhao, Feihu Zhang, Yu Ma, Liang Wang, and Ruigang Yang. Augmented LiDAR simulator for autonomous driving. *IEEE RA-L*, 2020.

[95] Jin Fang, Xinxin Zuo, Dingfu Zhou, Shengze Jin, Sen Wang, and Liangjun Zhang. LiDAR-Aug: A general rendering-based augmentation framework for 3D object detection. In *CVPR*, 2021.

[96] Qi Fang, Qing Shuai, Junting Dong, Hujun Bao, and Xiaowei Zhou. Reconstructing 3d human pose by watching humans in the mirror. In *CVPR*, 2021.

[97] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *ACM Communications*, 1981.

[98] Andrew W Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. In *ECCV*, 1998.

[99] Georgios Floros, Benito Van Der Zander, and Bastian Leibe. Openstreetslam: Global vehicle localization using openstreetmaps. In *ICRA*, 2013.

[100] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. *arXiv*, 2019.

[101] William T Freeman. The generic viewpoint assumption in a framework for visual perception. *Nature*, 1994.

[102] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[103] Hugo Germain, Vincent Lepetit, and Guillaume Bourmaud. Visual correspondence hallucination. *arXiv*, 2021.

[104] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016.

[105] Georgia Gkioxari, Alexander Toshev, and Navdeep Jaitly. Chained predictions using convolutional neural networks. In *ECCV*, 2016.

[106] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *ICCV*, 2019.

[107] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *ICCV*, 2009.

[108] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[109] Klaus Greff, Rupesh K Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. *arXiv*, 2016.

[110] Keith Grochow, Steven L Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. In *TOG*, 2004.

[111] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *ICML*, 2020.

[112] Roger Grosse, Micah K Johnson, Edward H Adelson, and William T Freeman. Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *ICCV*, 2009.

[113] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *CVPR*, 2018.

[114] Jiayuan Gu, Wei-Chiu Ma, Sivabalan Manivasagam, Wenyuan Zeng, Zihao Wang, Yuwen Xiong, Hao Su, and Raquel Urtasun. Weakly-supervised 3d shape completion in the wild. In *ECCV*, 2020.

[115] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *CVPR*, 2018.

[116] Michelle Guo, Alireza Fathi, Jiajun Wu, and Thomas Funkhouser. Object-centric neural scene rendering. *arXiv*, 2020.

[117] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *ICLR*, 2016.

[118] Marc Habermann, Weipeng Xu, Michael Zollhofer, Gerard Pons-Moll, and Christian Theobalt. Deepcap: Monocular human performance capture using weak supervision. In *CVPR*, 2020.

[119] Bumsub Ham, Minsu Cho, Cordelia Schmid, and Jean Ponce. Proposal flow. In *CVPR*, 2016.

[120] Kai Han, Rafael S Rezende, Bumsub Ham, Kwan-Yee K Wong, Minsu Cho, Cordelia Schmid, and Jean Ponce. Scnet: Learning semantic correspondence. In *ICCV*, 2017.

[121] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2000.

[122] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[123] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. *arXiv*, 2022.

[124] Daniel Hauagge, Scott Wehrwein, Kavita Bala, and Noah Snavely. Photometric ambient occlusion. In *CVPR*, 2013.

[125] James Hays and Alexei A Efros. Im2gps: estimating geographic information from a single image. In *CVPR*, 2008.

[126] Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. *TPAMI*, 2010.

[127] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *TPAMI*, 2015.

[128] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[129] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[130] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995.

[131] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NeurIPS*, 2017.

[132] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *ACCV*, 2012.

[133] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *TPAMI*, 2007.

[134] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

[135] William Hoff and Narendra Ahuja. Surfaces from stereo: Integrating feature matching, disparity estimation, and contour detection. *TPAMI*, 1989.

[136] Berthold KP Horn. Determining lightness from an image. *Computer graphics and image processing*, 1974.

[137] Berthold KP Horn. Obtaining shape from shading information. *The psychology of computer vision*, 1975.

[138] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 1981.

[139] Bo Hu, Christopher Brown, and Randal Nelson. Multiple-view 3-d reconstruction using a mirror. 2005.

[140] Peiyun Hu and Deva Ramanan. Bottom-up and top-down reasoning with hierarchical rectified gaussians. In *CVPR*, 2016.

[141] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015.

[142] Frédéric Huguet and Frédéric Devernay. A variational method for scene flow estimation from stereo sequences. In *ICCV*, 2007.

[143] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *CVPR*, 2018.

[144] Wei-Chih Hung, Varun Jampani, Sifei Liu, Pavlo Molchanov, Ming-Hsuan Yang, and Jan Kautz. Scops: Self-supervised co-part segmentation. In *CVPR*, 2019.

[145] Jeffrey Ichnowski, Yahav Avigal, Justin Kerr, and Ken Goldberg. Dex-nerf: Using a neural radiance field to grasp transparent objects. *arXiv*, 2021.

[146] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017.

[147] Eddy Ilg, Tonmoy Saikia, Margret Keuper, and Thomas Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *ECCV*, 2018.

[148] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[149] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *NeurIPS*, 2015.

[150] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv*, 2016.

[151] Michael Janner, Jiajun Wu, Tejas D Kulkarni, Ilker Yildirim, and Josh Tenenbaum. Self-supervised intrinsic image decomposition. In *NeurIPS*, 2017.

[152] Sebastian Hoppe Nesgaard Jensen, Mads Emil Brix Doest, Henrik Aanæs, and Alessio Del Bue. A benchmark and evaluation of non-rigid structure from motion. *IJCV*, 2021.

[153] Linyi Jin, Shengyi Qian, Andrew Owens, and David F Fouhey. Planar surface reconstruction from sparse views. 2021.

[154] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[155] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *ICCV*, 2015.

[156] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Scott Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social interaction capture. *TPAMI*, 2017.

[157] Hanbyul Joo, Tomas Simon, and Yaser Sheikh. Total capture: A 3d deformation model for tracking faces, hands, and bodies. In *CVPR*, 2018.

[158] Hanbyul Joo, Natalia Neverova, and Andrea Vedaldi. Exemplar fine-tuning for 3d human pose fitting towards in-the-wild 3d human pose estimation. In *3DV*, 2021.

[159] B. Julesz. Visual pattern discrimination. *IRE Transactions on Information Theory*, 1962.

[160] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 1986.

[161] Takeo Kanade and Masatoshi Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *TPAMI*, 1994.

[162] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *CVPR*, 2018.

[163] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018.

[164] Brian Karis and Epic Games. Real shading in unreal engine 4. *Proc. Physically Based Shading Theory Practice*, 2013.

[165] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv*, 2017.

[166] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[167] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. *TOG*, 2011.

[168] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *CVPR*, 2018.

[169] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, 2015.

[170] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *ICCV*, 2017.

[171] Seung Wook Kim, Jonah Philion, Antonio Torralba, and Sanja Fidler. DriveGAN: Towards a controllable high-quality neural simulation. In *CVPR*, 2021.

[172] Seungryong Kim, Dongbo Min, Bumsub Ham, Sangryul Jeon, Stephen Lin, and Kwanghoon Sohn. Fcss: Fully convolutional self-similarity for dense semantic correspondence. In *CVPR*, 2017.

[173] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[174] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv*, 2013.

[175] Jing Yu Koh, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson. Pathdreamer: A world model for indoor navigation. 2021.

[176] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *ICCV*, 2019.

[177] Chen Kong and Simon Lucey. Deep non-rigid structure from motion. In *ICCV*, October 2019.

[178] Kuldeep Kulkarni, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok. Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *CVPR*, 2016.

[179] Nilesh Kulkarni, Ishan Misra, Shubham Tulsiani, and Abhinav Gupta. 3d-relnet: Joint object and relational network for 3d prediction. In *ICCV*, 2019.

[180] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On measuring the accuracy of slam algorithms. *Autonomous Robots*, 2009.

[181] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *CVPR*, 2022.

[182] Pierre-Yves Laffont and Jean-Charles Bazin. Intrinsic decomposition of image sequences from local temporal variations. In *ICCV*, 2015.

[183] Pierre-Yves Laffont, Adrien Bousseau, and George Drettakis. Rich intrinsic image decomposition of outdoor scenes from multiple views. *IEEE transactions on visualization and computer graphics*, 2012.

[184] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, 2017.

[185] Edwin H Land and John J McCann. Lightness and retinex theory. *Josa*, 1971.

[186] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[187] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[188] Hendrik PA Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatial appearance and geometric detail. *TOG*, 2003.

[189] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *ICCV*, 2011.

[190] Anat Levin. Blind motion deblurring using image statistics. In *NeurIPS*, 2007.

[191] Jesse Levinson and Sebastian Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *ICRA*, 2010.

[192] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. Map-based precision vehicle localization in urban environments. In *RSS*, 2007.

[193] Wei Li, CW Pan, Rong Zhang, JP Ren, YX Ma, Jin Fang, FL Yan, QC Geng, XY Huang, HJ Gong, et al. Aads: Augmented autonomous driving simulation using data-driven algorithms. *Science robotics*, 2019.

[194] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *ECCV*, 2018.

[195] Yikai Li, Jiayuan Mao, Xiuming Zhang, Bill Freeman, Josh Tenenbaum, Noah Snavely, and Jiajun Wu. Multi-plane program induction with 3d box priors. *NeurIPS*, 2020.

[196] Yikai Li, Jiayuan Mao, Xiuming Zhang, William T Freeman, Joshua B Tenenbaum, and Jiajun Wu. Perspective plane program induction from a single image. In *CVPR*, 2020.

[197] Yunpeng Li, Noah Snavely, Dan Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3d point clouds. In *ECCV*, 2012.

[198] Zhengqi Li and Noah Snavely. Learning intrinsic image decomposition from watching the world. In *CVPR*, 2018.

[199] Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T Freeman. Learning the depths of moving people by watching frozen people. In *CVPR*, 2019.

[200] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *ECCV*, 2022.

[201] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *CVPR*, 2015.

[202] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *arXiv*, 2021.

[203] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *PAMI*, 2022.

[204] Daniel Lichy, Jiaye Wu, Soumyadip Sengupta, and David W Jacobs. Shape and material capture at home. In *CVPR*, 2021.

[205] Chen-Hsuan Lin and Simon Lucey. Inverse compositional spatial transformer networks. In *CVPR*, 2017.

[206] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *AAAI*, 2018.

[207] Chen-Hsuan Lin, Ersin Yumer, Oliver Wang, Eli Shechtman, and Simon Lucey. St-gan: Spatial transformer generative adversarial networks for image compositing. In *CVPR*, 2018.

[208] Chen-Hsuan Lin, Oliver Wang, Bryan C Russell, Eli Shechtman, Vladimir G Kim, Matthew Fisher, and Simon Lucey. Photometric mesh optimization for video-aligned 3d object reconstruction. In *CVPR*, 2019.

[209] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. Sdf-srn: Learning signed distance 3d object reconstruction from static images. *NeurIPS*, 2020.

[210] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. *arXiv*, 2021.

[211] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. *arXiv*, 2021.

[212] Chieh Hubert Lin, Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, and Ming-Hsuan Yang. InfinityGAN: Towards infinite-pixel image synthesis. In *ICLR*, 2022.

[213] Yen-Chen Lin, Pete Florence, Andy Zeng, Jonathan T Barron, Yilun Du, Wei-Chiu Ma, Anthony Simeonov, Alberto Rodriguez Garcia, and Phillip Isola. Mira: Mental imagery for robotic affordances. In *CoRL*, 2023.

[214] Zhi-Hao Lin, Wei-Chiu Ma, Hao-Yu Hsu, Yu-Chiang Frank Wang, and Shenlong Wang. Neurmips: Neural mixture of planar experts for view synthesis. In *CVPR*, 2022.

[215] Chris Linegar, Winston Churchill, and Paul Newman. Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained localisation. In *ICRA*, 2015.

[216] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *ICCV*, 2021.

[217] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *ECCV*, 2018.

[218] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020.

[219] Liu Liu, Hongdong Li, and Yuchao Dai. Efficient global 2d-3d matching for camera localization in a large-scale 3d map. In *ICCV*, 2017.

[220] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. *arXiv*, 2019.

[221] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *ICCV*, 2021.

[222] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[223] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. In *SIGGRAPH*, 2019.

[224] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 1981.

[225] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *SIGGRAPH Asia*, 2015.

[226] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 2004.

[227] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *CVPR*, 2016.

[228] Zhaoyang Lv, Chris Beall, Pablo F Alcantarilla, Fuxin Li, Zsolt Kira, and Frank Dellaert. A continuous optimization approach for efficient and accurate scene flow. In *ECCV*, 2016.

[229] Wei-Chiu Ma, Shenlong Wang, Marcus A Brubaker, Sanja Fidler, and Raquel Urtasun. Find your way by observing the sun and other semantic cues. In *ICRA*, 2017.

[230] Wei-Chiu Ma, Hang Chu, Bolei Zhou, Raquel Urtasun, and Antonio Torralba. Single image intrinsic decomposition without a single intrinsic image. In *ECCV*, 2018.

[231] Wei-Chiu Ma, Ignacio Tartavull, Ioan Andrei Bârsan, Shenlong Wang, Min Bai, Gellert Mattyus, Namdar Homayounfar, Shrinidhi Kowshika Lakshmikanth, Andrei Pokrovsky, and Raquel Urtasun. Exploiting sparse semantic hd maps for self-driving vehicle localization. In *IROS*, 2019.

[232] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *CVPR*, 2019.

[233] Yi Ma, Stefano Soatto, Jana Kosecka, and S Shankar Sastry. *An invitation to 3-d vision: from images to geometric models*. Springer Science & Business Media, 2012.

[234] Alexander Majercik, Cyril Crassin, Peter Shirley, and Morgan McGuire. A ray-box intersection algorithm and efficient dynamic voxel rendering. *Journal of Computer Graphics Techniques Vol*, 2018.

[235] Jitendra Malik and Pietro Perona. Preattentive texture discrimination with early vision mechanisms. 1990.

[236] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6d pose refinement in rgb. In *ECCV*, 2018.

[237] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *CVPR*, 2020.

[238] Stephen Robert Marschner. *Inverse rendering for computer graphics*. Cornell University, 1998.

[239] Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation. *TOG*, 2021.

[240] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *CVPR*, 2017.

[241] Kevin Matzen and Noah Snavely. Nyc3dcars: A dataset of 3d vehicles in geographic context. In *ICCV*, 2013.

[242] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016.

[243] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, 2015.

[244] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019.

[245] Tomer Michaeli and Michal Irani. Blind deblurring using internal patch recurrence. In *ECCV*, 2014.

[246] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[247] Frank Moosmann and Christoph Stiller. Joint self-localization and tracking of generic objects in 3d range data. In *ICRA*, 2013.

[248] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. 2022.

[249] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *CVPR*, 2022.

[250] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *T-RO*, 2017.

[251] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*, 2017.

[252] Nelson Nauata, Kai-Hung Chang, Chin-Yi Cheng, Greg Mori, and Yasutaka Furukawa. House-gan: Relational generative adversarial networks for graph-constrained house layout generation. In *ECCV*, 2020.

[253] Nelson Nauata, Sepidehsadat Hosseini, Kai-Hung Chang, Hang Chu, Chin-Yi Cheng, and Yasutaka Furukawa. House-gan++: Generative adversarial layout refinement networks. *arXiv*, 2021.

[254] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, Chakravarty R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. In *Computer Graphics Forum*, 2021.

[255] Peter Nelson, Winston Churchill, Ingmar Posner, and Paul Newman. From dusk till dawn: Localisation at night using artificial light sources. In *ICRA*, 2015.

[256] Michal Neoral and Jan Šochman. Object scene flow with temporal consistency. In *CVWW*, 2017.

[257] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[258] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *ICCV*, 2019.

[259] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, 2021.

[260] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

[261] David Novotny, Roman Shapovalov, and Andrea Vedaldi. Canonical 3D Deformer Maps: Unifying parametric and non-parametric methods for dense weakly-supervised category reconstruction. In *NeurIPS*, 2020.

[262] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Training a feedback loop for hand pose estimation. In *ICCV*, 2015.

[263] Byong Mok Oh, Max Chen, Julie Dorsey, and Frédo Durand. Image-based modeling and photo editing. In *SIGGRAPH*, 2001.

[264] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: Learning local features from images. *arXiv*, 2018.

[265] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *CVPR*, 2021.

[266] Julian Ost, Issam Laradji, Alejandro Newell, Yuval Bahat, and Felix Heide. Neural point light fields. In *CVPR*, 2022.

[267] Jinshan Pan, Zhe Hu, Zhixun Su, and Ming-Hsuan Yang. l_0-regularized intensity and gradient prior for deblurring text images and beyond. *TPAMI*, 2016.

[268] Jinshan Pan, Deqing Sun, Hanspeter Pfister, and Ming-Hsuan Yang. Blind image deblurring using dark channel prior. In *CVPR*, 2016.

[269] Nils Papenberg, Andrés Bruhn, Thomas Brox, Stephan Didas, and Joachim Weickert. Highly accurate optic flow computation with theoretically justified warping. *IJCV*, 2006.

[270] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. In *NeurIPS*, 2021.

[271] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[272] Georgios Pavlakos, Jitendra Malik, and Angjoo Kanazawa. Human mesh recovery from multiple shots. *arXiv*, 2020.

[273] Dario Pavllo, David Grangier, and Michael Auli. Quaternet: A quaternion-based recurrent model for human motion. In *BMVS*, 2018.

[274] Chi-Han Peng, Yong-Liang Yang, and Peter Wonka. Computing layouts with deformable templates. *TOG*, 2014.

[275] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *IJCV*, 2004.

[276] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *IJCV*, 2007.

[277] Javier Portilla, Vasily Strela, Martin J Wainwright, and Eero P Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *TIP*, 2003.

[278] Jens Puwein, Luca Ballan, Remo Ziegler, and Marc Pollefeys. Joint camera pose estimation and 3d human pose estimation in a multi-camera setup. In *ACCV*, 2014.

[279] Shengyi Qian, Linyi Jin, and David F Fouhey. Associative3d: Volumetric reconstruction from sparse views. In *ECCV*, 2020.

[280] Shengyi Qian, Alexander Kirillov, Nikhila Ravi, Devendra Singh Chaplot, Justin Johnson, David F Fouhey, and Georgia Gkioxari. Recognizing scenes from novel viewpoints. *arXiv*, 2021.

[281] Yiming Qian, Hao Zhang, and Yasutaka Furukawa. Roof-gan: learning to generate roof geometry and relations for residential houses. In *CVPR*, 2021.

[282] Xiaozhi Qu, Bahman Soheilian, and Nicolas Paparoditis. Vehicle localization using mono-camera and geo-referenced traffic signs. In *IV*, 2015.

[283] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv*, 2015.

[284] Ruslan Rakhimov, Andrei-Timotei Ardelean, Victor Lempitsky, and Evgeny Burnaev. Npbg++: Accelerating neural point-based graphics. In *CVPR*, 2022.

[285] Varun Ramakrishna, Daniel Munoz, Martial Hebert, James Andrew Bagnell, and Yaser Sheikh. Pose machines: Articulated pose estimation via inference machines. In *ECCV*, 2014.

[286] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *TPAMI*, 2020.

[287] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021.

[288] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, 2017.

[289] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Pytorch3d. `https://github.com/facebookresearch/pytorch3d`, 2020.

[290] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *NeurIPS*, 2019.

[291] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. In *CVPR*, 2021.

[292] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs. In *ICCV*, 2021.

[293] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *CVPR*, 2022.

[294] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015.

[295] Zhile Ren, Deqing Sun, Jan Kautz, and Erik Sudderth. Cascaded scene flow prediction using semantic segmentation. In *3DV*, 2017.

[296] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015.

[297] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

[298] JH Rick Chang, Chun-Liang Li, Barnabas Poczos, BVK Vijaya Kumar, and Aswin C Sankaranarayanan. One network to solve them all–solving linear inverse problems using deep projection models. In *ICCV*, 2017.

[299] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *ECCV*, 2020.

[300] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *CVPR*, 2021.

[301] Chris Rockwell, David F. Fouhey, and Justin Johnson. Pixelsynth: Generating a 3d-consistent experience from a single image. In *ICCV*, 2021.

[302] Lukasz Romaszko, Christopher K.I. Williams, and John Winn. Learning direct optimization for scene understanding. *Pattern Recognition*, 2020.

[303] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3d priors. In *ICCV*, 2021.

[304] Carsten Rother, Martin Kiefel, Lumin Zhang, Bernhard Schölkopf, and Peter V Gehler. Recovering intrinsic images with a global sparsity prior on reflectance. In *NeurIPS*, 2011.

[305] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, pages 2564–2571, 2011.

[306] Abbas Sadat, Mengye Ren, Andrei Pokrovsky, Yen-Chen Lin, Ersin Yumer, and Raquel Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. *IROS*, 2019.

[307] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020.

[308] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020.

[309] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Illumination from shadows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003.

[310] Yoichi Sato, Mark D Wheeler, and Katsushi Ikeuchi. Object shape and reflectance modeling from observation. In *SIGGRAPH*, 1997.

[311] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *ICCV*, 2011.

[312] Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. *NeurIPS*, 2021.

[313] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002.

[314] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016.

[315] Johannes L Schönberger, Marc Pollefeys, Andreas Geiger, and Torsten Sattler. Semantic visual localization. In *CVPR*, 2018.

[316] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016.

[317] Markus Schreiber, Carsten Knöppel, and Uwe Franke. Laneloc: Lane marking based localization using highly accurate maps. In *IV*, 2013.

[318] Mohammad Amin Shabani, Weilian Song, Makoto Odamaki, Hirochika Fujiki, and Yasutaka Furukawa. Extreme structure from motion for indoor panoramas without visual overlaps. In *ICCV*, 2021.

[319] Yuan Shen, Wei-Chiu Ma, and Shenlong Wang. SGAM: Building a virtual 3d world through simultaneous generation and mapping. In *NeurIPS*, 2022.

[320] Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning. In *CVPR*, 2018.

[321] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *CVPR*, 2013.

[322] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[323] Sudipta N Sinha and Marc Pollefeys. Camera network calibration and synchronization from silhouettes in archived video. *IJCV*, 2010.

[324] Sudipta N Sinha, Marc Pollefeys, and Leonard McMillan. Camera network calibration from dynamic silhouettes. In *CVPR*, 2004.

[325] Ruben M Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. A survey on procedural modelling for virtual worlds. In *Computer Graphics Forum*, 2014.

[326] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH*. 2006.

[327] Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *IJCV*, 2008.

[328] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv*, 2020.

[329] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. NeRV: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021.

[330] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022.

[331] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010.

[332] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018.

[333] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of cnns for optical flow estimation. *TPAMI*, 2019.

[334] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *CVPR*, 2021.

[335] Chris Sweeney, Aleksander Holynski, Brian Curless, and Steve M Seitz. Structure from motion for panorama-style videos. *arXiv*, 2019.

[336] Richard Szeliski and Sing Bing Kang. Recovering 3d shape and motion from image streams using nonlinear least squares. *Journal of Visual Communication and Image Representation*, 1994.

[337] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *CVPR*, 2021.

[338] Abhijeet Tallavajhula, Çetin Meriçli, and Alonzo Kelly. Off-road lidar simulation with data-driven terrain primitives. In *ICRA*, 2018.

[339] Shuhan Tan, Kelvin Wong, Shenlong Wang, Sivabalan Manivasagam, Mengye Ren, and Raquel Urtasun. Scenegen: Learning to generate realistic traffic scenes. In *CVPR*, 2021.

[340] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable large scene neural view synthesis. In *CVPR*, 2022.

[341] Johan WH Tangelder and Remco C Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia tools and applications*, 2008.

[342] Tatsunori Taniai, Sudipta N Sinha, and Yoichi Sato. Fast multi-frame stereo scene flow with motion segmentation. In *CVPR*, 2017.

[343] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *CVPR*, 2019.

[344] Piotr Teterwak, Aaron Sarna, Dilip Krishnan, Aaron Maschinot, David Belanger, Ce Liu, and William T Freeman. Boundless: Generative adversarial networks for image extension. In *ICCV*, 2019.

[345] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *TPAMI*, 2009.

[346] Philip HS Torr and Andrew Zisserman. Feature based methods for structure and motion estimation. In *International workshop on vision algorithms*, 1999.

[347] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014.

[348] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. *International workshop on vision algorithms*, 1999.

[349] Zhuowen Tu. Auto-context and its application to high-level vision tasks. In *CVPR*, 2008.

[350] Hsiao-Yu Tung, Hsiao-Wei Tung, Ersin Yumer, and Katerina Fragkiadaki. Self-supervised learning of motion capture. In *NeurIPS*, 2017.

[351] Hsiao-Yu Fish Tung, Adam W Harley, William Seto, and Katerina Fragkiadaki. Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. In *ICCV*, 2017.

[352] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *CVPR*, 2018.

[353] Levi Valgaerts, Andrés Bruhn, Henning Zimmer, Joachim Weickert, Carsten Stoll, and Christian Theobalt. Joint estimation of motion, structure and geometry from stereo sequences. In *ECCV*, 2010.

[354] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *NeurIPS*, 2017.

[355] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *ICCV*, 1999.

[356] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *CVPR*, 2022.

[357] Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. Neural kinematic networks for unsupervised motion retargetting. In *CVPR*, 2018.

[358] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *ICCV*, 2013.

[359] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a piecewise rigid scene model. *ICCV*, 2015.

[360] Chaoyang Wang, Chen-Hsuan Lin, and Simon Lucey. Deep nrsfm++: Towards unsupervised 2d-3d lifting in the wild. In *3DV*, 2020.

[361] Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. In *CVPR*, 2021.

[362] Jingkang Wang, Sivabalan Manivasagam, Yun Chen, Ze Yang, Ioan Andrei Bârsan, Anqi Joyce Yang, Wei-Chiu Ma, and Raquel Urtasun. CADSim: Robust and scalable in-the-wild 3d reconstruction for controllable sensor simulation. In *Conference on Robot Learning*, 2022.

[363] Jingkang Wang, Sivabalan Manivasagam, Yun Chen, Ze Yang, Ioan Andrei Bârsan, Anqi Joyce Yang, Wei-Chiu Ma, and Raquel Urtasun. Cadsim: Robust and scalable in-the-wild 3d reconstruction for controllable sensor simulation. In *CoRL*, 2022.

[364] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021.

[365] Qianqian Wang, Xiaowei Zhou, Bharath Hariharan, and Noah Snavely. Learning feature descriptors using camera pose supervision. In *ECCV*, volume 12346, 2020.

[366] Shenlong Wang, Sanja Fidler, and Raquel Urtasun. Holistic 3d scene understanding from a single geo-tagged image. In *CVPR*, 2015.

[367] Shenlong Wang, Sanja Fidler, and Raquel Urtasun. Lost shopping! monocular localization in large indoor spaces. In *ICCV*, 2015.

[368] Shenlong Wang, Linjie Luo, Ning Zhang, and Jia Li. Autoscaler: scale-attention networks for visual correspondence. *arXiv*, 2016.

[369] Shenlong Wang, Linjie Luo, Ning Zhang, and Jia Li. Autoscaler: Scale-attention networks for visual correspondence. *arXiv*, 2016.

[370] Shenlong Wang, Sean Ryan Fanello, Christoph Rhemann, Shahram Izadi, and Pushmeet Kohli. The global patch collider. In *CVPR*, 2016.

[371] Shenlong Wang, Min Bai, Gellert Mattyus, Hang Chu, Wenjie Luo, Bin Yang, Justin Liang, Joel Cheverie, Sanja Fidler, and Raquel Urtasun. Torontocity: Seeing the world with a million eyes. In *ICCV*, 2017.

[372] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[373] Tsun-Hsuan Wang, Alexander Amini, Wilko Schwarting, Igor Gilitschenski, Sertac Karaman, and Daniela Rus. Learning interactive driving policies via data-driven simulation. In *ICRA*, 2022.

[374] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019.

[375] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *ECCV*, 2018.

[376] Zhangyang Wang, Yingzhen Yang, Zhaowen Wang, Shiyu Chang, Jianchao Yang, and Thomas S Huang. Learning super-resolution jointly from external and internal examples. *TIP*, 2015.

[377] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004.

[378] Zian Wang, Wenzheng Chen, David Acuna, Jan Kautz, and Sanja Fidler. Neural light field estimation for street scenes with differentiable virtual object insertion. *ECCV*, 2022.

[379] Zian Wang, Tianchang Shen, Jun Gao, Shengyu Huang, Jacob Munkberg, Jon Hasselgren, Zan Gojcic, Wenzheng Chen, and Sanja Fidler. Neural fields meet explicit geometric representation for inverse rendering of urban scenes. *arXiv*, 2023.

[380] Jan D Wegner, Steven Branson, David Hall, Konrad Schindler, and Pietro Perona. Cataloging public objects using aerial and street-level images-urban trees. In *CVPR*, 2016.

[381] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[382] David Weiss and Benjamin Taskar. Structured prediction cascades. In *AISTATS*, 2010.

[383] Andre Welzel, Pierre Reisdorf, and Gerd Wanielik. Improving urban vehicle localization with traffic sign recognition. In *ICITS*, 2015.

[384] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. *arXiv*, 2019.

[385] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *CVPR*, 2020.

[386] Ryan W Wolcott and Ryan M Eustice. Visual localization within lidar maps for automated urban driving. In *IROS*, 2014.

[387] Ryan W Wolcott and Ryan M Eustice. Fast lidar localization using multiresolution gaussian mixture maps. In *ICRA*, 2015.

[388] Changchang Wu et al. Visualsfm: A visual structure from motion system. 2011.

[389] Jiajun Wu, Joseph J Lim, Hongyi Zhang, and Joshua B Tenenbaum. Physics 101: Learning physical object properties from unlabeled videos.

[390] Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *NeurIPS*, 2015.

[391] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *NeurIPS*, 2016.

[392] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. *NeurIPS*, 2017.

[393] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, et al. Pandaset: Advanced sensor suite dataset for autonomous driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021.

[394] Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *CVPR*, 2013.

[395] Li Xu, Jimmy SJ Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. In *NeurIPS*, 2014.

[396] Yan Xu, Yu-Jhe Li, Xinshuo Weng, and Kris Kitani. Wide-baseline multi-camera calibration using person re-identification. In *CVPR*, June 2021.

[397] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *ECCV*, 2014.

[398] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *ICCV*, 2019.

[399] Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. Geometry processing with neural fields. *NeurIPS*, 2021.

[400] Ze Yang, Siva Manivasagam, Ming Liang, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Recovering and simulating pedestrians in the wild. In *CoRL*, 2020.

[401] Ze Yang, Shenlong Wang, Siva Manivasagam, Zeng Huang, Wei-Chiu Ma, Xinchen Yan, Ersin Yumer, and Raquel Urtasun. S3: Neural shape, skeleton, and skinning fields for 3d human modeling. In *CVPR*, 2021.

[402] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. *CVPR*, 2023.

[403] Ze Yang, Sivabalan Manivasagam, Yun Chen, Jingkang Wang, Rui Hu, and Raquel Urtasun. Reconstructing objects in-the-wild for realistic sensor simulation. In *ICRA*, 2023.

[404] Zhenpei Yang, Jeffrey Z Pan, Linjie Luo, Xiaowei Zhou, Kristen Grauman, and Qixing Huang. Extreme relative pose estimation for rgb-d scans via scene completion. In *CVPR*, 2019.

[405] Zhenpei Yang, Yuning Chai, Dragomir Anguelov, Yin Zhou, Pei Sun, Dumitru Erhan, Sean Rafferty, and Henrik Kretzschmar. Surfelgan: Synthesizing realistic sensor data for autonomous driving. In *CVPR*, 2020.

[406] Zhenpei Yang, Siming Yan, and Qixing Huang. Extreme relative pose network under hybrid representations. In *CVPR*, 2020.

[407] Shunyu Yao, Tzu Ming Hsu, Jun-Yan Zhu, Jiajun Wu, Antonio Torralba, Bill Freeman, and Josh Tenenbaum. 3d-aware scene manipulation via inverse graphics. In *NeurIPS*, 2018.

[408] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *NeurIPS*, 2020.

[409] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. *arXiv*, 2018.

[410] Keisuke Yoneda, Hossein Tehrani, Takashi Ogawa, Naohisa Hukuyama, and Seiichi Mita. Lidar scan feature for localization with highly precise 3-d map. In *IV*, 2014.

[411] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *SIGGRAPH*, 1999.

[412] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. In *NeurIPS*, 2022.

[413] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015.

[414] Amir R Zamir, Asaad Hakeem, Luc Van Gool, Mubarak Shah, Richard Szeliski, AR Zamir, A Hakeem, L VanGool, M Shah, and R Szeliski. Large-scale visual geo-localization preface. 2016.

[415] Amir R Zamir, Te-Lin Wu, Lin Sun, William B Shen, Bertram E Shi, Jitendra Malik, and Silvio Savarese. Feedback networks. In *CVPR*, 2017.

[416] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *CVPR*, 2015.

[417] Jure Zbontar, Yann LeCun, et al. Stereo matching by training a convolutional neural network to compare image patches. *JMLR*, 2016.

[418] Albert J Zhai and Shenlong Wang. Peanut: Predicting and navigating to unseen targets. *arXiv*, 2022.

[419] Chris Zhang, Wenjie Luo, and Raquel Urtasun. Efficient convolutions for real-time semantic segmentation of 3d point clouds. In *3DV*, 2018.

[420] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *RSS*, 2014.

[421] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. In *ICCV*, 2019.

[422] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *TIP*, 2017.

[423] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NeRF++: Analyzing and improving neural radiance fields. *arXiv*, 2020.

[424] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *CVPR*, 2021.

[425] Kai Zhang, Fujun Luan, Zhengqi Li, and Noah Snavely. Iron: Inverse rendering by optimizing neural sdfs and materials from photometric images. In *CVPR*, 2022.

[426] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. *IEEE*, 2018.

[427] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[428] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *TOG*, 2021.

[429] Xuaner Zhang, Ren Ng, and Qifeng Chen. Single image reflection separation with perceptual losses. In *CVPR*, 2018.

[430] Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. Modeling indirect illumination for inverse rendering. In *CVPR*, 2022.

[431] Yunzhi Zhang, Shangzhe Wu, Noah Snavely, and Jiajun Wu. Seeing a rose in five thousand ways. *arXiv*, 2022.

[432] Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image gans meet differentiable rendering for inverse graphics and interpretable 3d neural rendering. *arXiv*, 2020.

[433] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[434] Kaiyang Zhou and Tao Xiang. Torchreid: A library for deep learning person re-identification in pytorch. *arXiv*, 2019.

[435] Tinghui Zhou, Yong Jae Lee, Stella X Yu, and Alyosha A Efros. Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. In *CVPR*, 2015.

[436] Tinghui Zhou, Philipp Krahenbuhl, and Alexei A Efros. Learning data-driven reflectance priors for intrinsic image decomposition. In *ICCV*, 2015.

[437] Tinghui Zhou, Philipp Krahenbuhl, Mathieu Aubry, Qixing Huang, and Alexei A Efros. Learning dense correspondence via 3d-guided cycle consistency. In *CVPR*, 2016.

[438] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, 2019.

[439] Song Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *IJCV*, 1998.

[440] Julius Ziegler, Henning Lategahn, Markus Schreiber, Christoph G Keller, Carsten Knöppel, Jochen Hipp, Martin Haueis, and Christoph Stiller. Video based localization for bertha. In *IV*, 2014.

[441] Maria Zontak and Michal Irani. Internal statistics of a single natural image. In *CVPR*. IEEE, 2011.

[442] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic lidar point clouds. In *ECCV*, 2022.