



INDRAPRASTHA INSTITUTE OF INFORMATION  
TECHNOLOGY DELHI

DOCTORAL THESIS

---

# Robust Scene Understanding

---

*Author:*  
Lokender Tiwari

*Supervisor:*  
Dr. Saket Anand

*A thesis submitted in fulfillment of the requirements  
for the degree of Doctor of Philosophy*

*in the*

Department of Computer Science and  
Engineering



# Certificate

This is to certify that the thesis titled, “Robust Scene Understanding” being submitted by *Lokender Tiwari* to the **Indraprastha Institute of Information Technology Delhi**, for the award of the degree of Doctor of Philosophy, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standard fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree or diploma.

Dr. Saket Anand  
Department of Computer Science and Engineering  
**Indraprastha Institute of Information Technology Delhi**  
New Delhi, 110020



————— Sanskrit —————

उत्साहो बलवानार्य नास्त्रयुत्साहात् परं बलम् ।  
सोत्साहस्य च लोकेषु न किञ्चिदपि दुर्लभम् ॥

————— Hindi —————

उत्साह बलवान् है । उत्साह से बढ़कर कोई बल नहीं है ।  
उत्साह हो तो समस्त विश्व में कुछ भी असंभव नहीं है ।

————— English —————

Enthusiasm is might, oh, noble one, there is no superior might than enthusiasm  
and to him with willingness, there is no impossibility in the world,... even the slightest...

– Ramayana, Kishkindhakand



# Abstract

Lokender Tiwari

## *Robust Scene Understanding*

A scene can be interpreted from two perspectives: *geometric* and *semantic*. Geometric scene understanding requires inferring the 3D layout of the scene from an image or video, while semantic scene understanding requires identification of the type of objects and their relationships. It is well known that geometric estimation is sensitive to random noise and outliers in the data and typically leverage robust estimation methods. Thanks to deep learning (DL) approaches, semantic scene understanding has seen substantial progress through advances in image classification, object detection, and semantic segmentation type tasks even in complex, cluttered scenes. However, DL techniques are known to be susceptible to carefully crafted noisy samples, popularly known as adversarial examples. Perceptually both clean and noisy/adversarial samples look very similar; even a human find it difficult to differentiate between them semantically. Essentially, the goal of an adversary is to add sufficient noise to the clean sample so that it doesn't follow the underlying classifier model learned in the form of neural network weights. This vulnerability has inspired the investigation of robust methods for deep neural networks as well.

The ultimate goal of an intelligent visual perception system is to mimic the human level scene understanding and reasoning from the images and videos. Collectively both semantic and geometric understanding plays a vital role in bridging the gap between human and machine vision's capability. The universal existence of noisy and corrupted input data/observations and the sensitivity of both geometric and semantic tasks towards them poses an important question on the reliability of such tasks in security-critical applications. For example, an autonomous car navigating in a city incorrectly classifies the red light into green or inaccurately estimates the distance from an obstacle, etc. Such an event leads to a catastrophic situation. What makes human perception unique is its robustness. Therefore, to mimic the human-level understanding, we believe that all the methods intended for scene understanding tasks must consider *robustness* as one of their prime and the *necessary* component, and evaluate their effectiveness on the same.

In this dissertation, we investigate the robustness of two *geometric* tasks and one *semantic* task towards robust scene understanding. *Geometric*: Multiple Geometric Model Fitting, Simultaneous Localization and Mapping (SLAM) using a monocular camera. *Semantic*: Robust Image Classification. We propose robust solutions to these three important scene understanding tasks.





# Acknowledgements

My Ph.D. journey was full of unexpected events. It had many ups and downs. Apart from research and technical skills, it has taught me valuable lessons that would help me in life. It has taught me how to handle unexpected situations and handle multiple tasks while maintaining a balance between professional and personal life. The journey would not have been possible without the support of many people. I take this opportunity to express my sincere gratitude to all those who made this Ph.D. thesis possible.

First and foremost, I would like to express my special thanks to my adviser, Dr. Saket Anand, for believing in me and agreeing on guiding me as his first Ph.D. student. He has helped me to improve in both technical and non-technical aspects. Now, when I look back, from where I started and where I am right now, I can certainly say, I am much more confident and comfortable to drive any research project independently. If this is what we expect from a graduating Ph.D. student, then I believe we both have succeeded in our respective tasks. He is a great mentor and adviser, always supportive and ready to help.

This work would not have been possible without support from NEC Laboratories America. I acknowledge and express my gratitude to Dr. Quoc-Huy Tran for giving me the opportunity to work with excellent researchers at NEC Labs. Huy's guidance helped me a lot to grow in both technical and non-technical aspects. I am grateful to Prof. Manmohan Chandraker and Dr. Pan Ji for their continuous guidance. I have learned a lot from the brainstorming sessions and discussions with Huy, Manmohan, and Pan. I would also like to thank Ms. Nicole from the administrative department of NEC Labs for her continuous support.

I want to thank my Ph.D. monitoring committee members, Dr. Ojaswa Sharma and Dr. Chetan Arora, for their fruitful comments throughout my Ph.D. journey. I would also like to thank Prof. Subhasis Banerjee for his valuable feedback on my Ph.D. comprehensive exam. I have learned a lot from discussions with him. I express my gratitude to the admin staff of IIIT-D, especially Ms. Priti, for being extremely helpful in the fast resolution of all admin related matters.

I want to express my gratitude to my awesome friends, Dhananjay Kimothi, Dr. Parag Aggarwal, Dr. Ambuj Mehrish, Akshay Jain, Anil Sharma, Dr. Milan Jain, Anupriya Tuli, Sandeepika Sharma for being with me in ups and downs of life. Having friends like them is a blessing.

I express my deepest appreciation for my parents, Shashi and Suresh Tiwari, and my brother Anupam Tiwari for their unconditional love, support, and belief in me. They are the source of my motivation.



# Contents

<b>Certificate</b>	<b>iii</b>
<b>Abstract</b>	<b>vii</b>
<b>Acknowledgements</b>	<b>ix</b>
<b>Contents</b>	<b>xi</b>
<b>List of Figures</b>	<b>xv</b>
<b>List of Tables</b>	<b>xix</b>
<b>List of Abbreviations</b>	<b>xxiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Scene Understanding . . . . .	1
1.2 Towards Robust Scene Understanding . . . . .	5
1.3 Thesis Contributions . . . . .	10
1.3.1 Self-Supervised, Self-Improving geometric-CNN Framework for 3D Perception . . . . .	10
1.3.1.1 Our Contributions . . . . .	10
1.3.2 Robust Multiple Geometric Model Fitting . . . . .	11
1.3.2.1 Our Contributions . . . . .	12
1.3.3 Robust Image Classification . . . . .	14
1.3.3.1 Our Contributions . . . . .	14
1.4 Thesis Organization . . . . .	15
1.5 Outcomes of this Thesis . . . . .	16
1.5.1 Publications . . . . .	16
1.5.2 Patent Applications . . . . .	16
1.5.3 Awards . . . . .	16
1.5.4 Open Source Code and Project Pages . . . . .	17
<b>2 Self-Improving geometric-CNN Framework for 3D Perception</b>	<b>19</b>
2.1 Introduction . . . . .	19
2.1.1 Contributions . . . . .	22
2.2 Monocular SLAM and Depth Prediction Approaches . . . . .	23
2.2.1 Monocular SLAM Approaches . . . . .	23
2.2.2 Unsupervised Monocular Depth Prediction Approaches . . . . .	23
2.2.3 Approaches Using Depth to Improve Monocular SLAM . . . . .	23
2.2.4 Approaches Using SLAM to Improve Monocular Depth Prediction . . . . .	24

2.3	Self-Improving geometric-CNN Framework . . . . .	24
2.3.1	Pseudo RGB-D for Improving Monocular SLAM . . . . .	25
2.3.2	Monocular SLAM for Improving Depth Prediction . . . . .	26
2.3.3	Narrow and Wide Baseline Losses . . . . .	26
2.3.3.1	Symmetric Depth Transfer Loss . . . . .	26
2.3.3.2	Depth Consistency Loss . . . . .	27
2.3.3.3	Photometric Reconstruction Loss . . . . .	27
2.4	Experimental Analysis . . . . .	28
2.4.1	Datasets and Evaluation Metrics . . . . .	28
2.4.2	Implementation Details . . . . .	29
2.4.3	Monocular Depth Prediction / Refinement Evaluation . . . . .	30
2.4.3.1	Quantitative Depth Evaluation Results on KITTI Eigen’s Split Test Set . . . . .	30
2.4.3.2	Qualitative Depth Evaluation / Improvement Results . . . . .	31
2.4.3.3	Quantitative Depth Evaluation Results on KITTI Odometry Validation Set (Sequences 09 and 10) . . . . .	32
2.4.3.4	Quantitative Depth Evaluation Results on TUM RGB-D Sequences . . . . .	32
2.4.3.5	Qualitative Depth Evaluation Results on TUM RGB-D Sequences . . . . .	33
2.4.4	Monocular SLAM/Pose Refinement Evaluation . . . . .	33
2.4.4.1	Quantitative Pose Evaluation Results on KITTI Odometry Sequences 09 and 10 . . . . .	33
2.4.4.2	Quantitative Pose Evaluation Results on KITTI Odometry Test Set . . . . .	33
2.4.4.3	Qualitative Pose Evaluation Results on KITTI Odometry . . . . .	34
2.4.4.4	Comparison with State-Of-The-Art SLAM Methods . . . . .	34
2.4.4.5	KITTI Odometry Leaderboard Results . . . . .	35
2.4.4.6	Quantitative Pose Evaluation Results on TUM RGB-D Se- quences . . . . .	36
2.4.4.7	Qualitative Pose Evaluation Results on TUM RGB-D Se- quences . . . . .	37
2.5	Analysis of Self-Improving Loops . . . . .	37
2.6	Discussion . . . . .	39
<b>3</b>	<b>Robust Multiple Geometric Model Fitting</b> . . . . .	<b>41</b>
3.1	Introduction . . . . .	41
3.1.1	Contributions . . . . .	44
3.2	Related Work . . . . .	45
3.2.1	Guided Sampling Approaches . . . . .	45
3.2.2	Full Multi-Model Fitting Approaches . . . . .	46
3.2.3	Traditional Robust Model Fitting Methods . . . . .	47
3.3	Problem Statement . . . . .	47
3.4	DGSAC Methodology . . . . .	48
3.5	Density Guided Sampling and Consensus . . . . .	49
3.5.1	Hypothesis and Point preferences . . . . .	49
3.5.2	Kernel Residual Density (KRD) . . . . .	50

3.5.2.1	Kernel Residual Density Scores . . . . .	51
3.5.2.2	KRD based point preferences. . . . .	53
3.5.3	Advantages of Kernel Residual Density . . . . .	53
3.5.4	KRD Based Point Correlation . . . . .	53
3.5.4.1	Residual vs. KRD Based Point Correlation . . . . .	54
3.5.5	KRD Guided Hypotheses Generation (KDGS) . . . . .	55
3.5.6	Overview of the KDGS Algorithm . . . . .	56
3.5.7	Conditional Hypothesis Generation for a Data Point . . . . .	56
3.5.7.1	Potential good hypotheses of a data point . . . . .	56
3.5.7.2	Conditional inlier probability using potential good hypotheses . . . . .	57
3.5.7.3	Explanation score . . . . .	58
3.5.7.4	Conditional Sampling . . . . .	58
3.5.8	KDGS Algorithm Flow . . . . .	60
3.5.8.1	Termination Analysis of KDGS . . . . .	61
3.5.9	KRD Based Inlier Noise Scale Estimation . . . . .	61
3.5.10	Greedy Method/Optimization Based Model Selection . . . . .	63
3.5.10.1	Hypothesis Correlation using Spearman-Footrule . . . . .	64
3.5.10.2	Model Hypothesis Goodness Measure . . . . .	64
3.5.10.3	Greedy Model Selection (GMS) . . . . .	64
3.5.10.4	Optimization Based Model Selection . . . . .	65
3.5.10.5	Diagonal Penalty Matrix . . . . .	66
3.5.11	Point-to-Model Assignment . . . . .	67
3.6	Experimental Analysis . . . . .	67
3.6.1	Experimental Analysis of KDGS . . . . .	69
3.6.2	Evaluation of full DGSAC Pipeline . . . . .	72
3.6.2.1	Evaluation on Real Datasets . . . . .	72
3.6.2.2	Evaluation on Synthetic Datasets . . . . .	75
3.7	Discussion . . . . .	76
<b>4</b>	<b>Robust Image Classification</b> . . . . .	<b>79</b>
4.1	Introduction . . . . .	79
4.1.1	Contributions . . . . .	81
4.2	Defense Approaches to Adversarial Attacks . . . . .	81
4.2.1	Modify Training and Modify Inputs During Testing . . . . .	82
4.2.2	Modify Network / Network Add-ons . . . . .	82
4.2.3	Defenses for Large Scale Image Classification . . . . .	82
4.3	REGroup Methodology . . . . .	82
4.4	DNN Layers as Generative Classifiers . . . . .	84
4.4.1	Layerwise Neural Response Distributions . . . . .	84
4.4.2	Layerwise Generative Classifiers . . . . .	85
4.5	Robust Predictions with Rank Aggregation . . . . .	86
4.5.1	Rank Aggregation using Borda Count . . . . .	86
4.5.2	Hyperparameter Settings . . . . .	87
4.6	Experimental Analysis . . . . .	87
4.6.1	Adversarial Attacks . . . . .	87
4.6.1.1	Gradient-Based Attacks . . . . .	87

4.6.1.2	Gradient-Free Attacks . . . . .	88
4.6.2	Experimental Setup . . . . .	88
4.6.3	Performance on Gradient-Based Attacks . . . . .	89
4.6.3.1	Comparison with adversarial-training / fine-tuning . . . . .	89
4.6.3.2	Performance w.r.t PGD adversarial strength . . . . .	89
4.6.3.3	Performance on un-targeted attacks. . . . .	90
4.6.4	Performance on unrestricted, untargeted semantic manipulation attack. . . . .	91
4.6.4.1	Performance on targeted attacks . . . . .	92
4.6.4.2	Performance on PGD attack with High Confidence . . . . .	92
4.6.4.3	Performance on Physically Realizable Attacks . . . . .	92
4.6.5	Performance on Gradient-Free Attacks . . . . .	93
4.6.6	Analysis and Ablation Study . . . . .	94
4.6.6.1	Accuracy vs number of layers ( $k$ ) . . . . .	94
4.6.6.2	Effect of positive and negative pre-activation responses. . . . .	95
4.6.6.3	Inference time using REGroup . . . . .	95
4.7	Discussion . . . . .	96
<b>5</b>	<b>Conclusion and Future Directions</b>	<b>97</b>

# List of Figures

- 1.1 Some Autonomous Driving Tasks . . . . . 1
- 1.2 Augmented Reality (AR) Applications . . . . . 2
- 1.3 Some Important Scene Understanding Tasks . . . . . 4
- 1.4 (a) Monocular SLAM, (b) Triangulation, (c) Keypoint matches . . . . . 5
- 1.5 (a) Struct SLAM 1.5, (b) Vanishing points and horizon line . . . . . 6
- 1.6 State-of-the-art unsupervised monocular depth prediction method MonoDepth2 [56] fails to predict accurate depth values corresponding to the van. . . . . 7
- 1.7 Adversarial attack examples, where a classifier fails to predict the correct class . . . . . 8
- 1.8 Scenarios where multiple instances of the same model can be simultaneously present. (a) Multiple motions, where a motion model of an object is defined by the fundamental matrix estimated using the true keypoint correspondences extracted from an image pair, e.g., fundamental matrix estimated using the red keypoint correspondences defines the motion model of the bus. (b) Fit planes to the point cloud of an indoor scene and (c) Segment coins by fitting circle to each coin. . . . . 12
- 1.9 Planar surface segmentation by fitting multiple planar homographies to the keypoint correspondences. In (c), red keypoint correspondences are gross-outliers. The magenta keypoint correspondences act as pseudo-outliers for planar surface denoted by blue keypoint correspondences and vice-versa . 13
- 1.10 Steps in a typical multiple model fitting pipeline. An illustrative example multiple line fitting. . . . . 13
  
- 2.1 RGB SLAM vs. RGB-D SLAM Robustness and Accuracy. RGB ORB-SLAM2 [102] fails to track after sometime, on KITTI benchmark dataset [52] sequence 01, RGB-D ORB-SLAM2 [102] successfully build the 3D map without tracking failure. . . . . 20
- 2.2 State-of-the-art unsupervised monocular depth prediction method MonoDepth2 [56] fails to predict accurate depth values for both the farther away points (top 2 rows) and also nearby points (bottom row). . . . . 21
- 2.3 MonoDepth2 [56] pose network camera poses vs. pseudo RGBD-SLAM camera poses where depth(D) is from CNN. The pose network from [56] leads to significant drift. . . . . 21
- 2.4 **Overview of Our Self-Improving Framework.** It alternates between pose refinement (blue arrows; Sec. 2.3.1) and depth refinement (red arrows; Sec. 2.3.2). . . . . 25

2.5	<b>Narrow and Wide Baseline Losses.</b> Narrow baseline photometric and smoothness losses involve keyframe $\mathcal{I}_c$ and temporally <i>adjacent</i> frames $\mathcal{I}_{c-1}$ and $\mathcal{I}_{c+1}$ , and wide baseline symmetric depth transfer and depth consistency losses involve keyframe $\mathcal{I}_c$ and temporally <i>farther</i> keyframes $\mathcal{I}_{k1}$ and $\mathcal{I}_{k2}$ . Refer to the text below for details. . . . .	26
2.6	Qualitative depth evaluation results on KITTI Eigen’s split test set. . . . .	31
2.7	Qualitative depth improvement results in depth prediction of farther away scene points. . . . .	32
2.8	Qualitative depth evaluation results on TUM RGB-D sequences. . . . .	33
2.9	Qualitative pose evaluation results on KITTI Odometry sequences. Note that both RGB ORB-SLAM fails in (b) and both RGB-SLAM and pRGBD-Initial fail in (d). . . . .	36
2.10	Qualitative pose evaluation results on TUM RGB-D sequences. Note that RGB ORB-SLAM fails in (a). . . . .	37
2.11	Depth/Pose evaluation metrics w.r.t. self-improving loops. (a). Absolute Relative (Abs Rel) ( <i>lower is better</i> ) (b). Squared Relative (Sq Rel) ( <i>lower is better</i> ) (c). RMSE ( <i>lower is better</i> ) (d). RMSE Log ( <i>lower is better</i> ), (e). $a_1$ ( <i>higher is better</i> ), (f). $a_2$ ( <i>higher is better</i> ), (g). $a_3$ ( <i>higher is better</i> ) and (h) Absolute Trajectory Error (RMSE) ( <i>lower is better</i> ). Depth evaluation metrics in (a-g) are computed at different max depth caps ranging from 30-80 meters. . . . .	38
3.1	<i>Johnsons</i> , a multiple-homography fitting example from AdelaideRMF dataset [161]. Varying ground truth inlier noise scale and number of inliers. . . . .	42
3.2	Density Guided Sampling and Consensus (DGSAC) . . . . .	44
3.3	<b>DGSAC pipeline</b> consists of four major steps. Step 1: KRD guided hypothesis generation (Sec. 3.5.5). Step 2: Inlier noise scale estimation (Sec. 3.5.9). Step 3: Model selection (Sec. 3.5.10). Step 4: Point-to-Model assignment (Sec. 3.5.11). Here, we have shown an example of multiple homography estimation, the point membership in the output/ground truth is color coded. Irrespective of the fitting tasks (homography estimation, fundamental matrix estimation, vanishing point estimation, line, circle, and plane fitting), the four steps of the DGSAC pipeline remain the same. . . . .	48
3.4	(a) Ground-Truth Structure, (b) Density based ( <b>magenta</b> ) and Residual based ( <b>green</b> ) top-1 preference of 5 inliers of the ground-truth structure ( <b>blue</b> ). It can be seen, the density based top preference of all 5 inliers is referring to the hypotheses generated from its true dense structure, while residual based top-1 preferences are the arbitrary hypotheses to which these data points have small residuals only. (c) a good ( <b>magenta</b> ) and a bad ( <b>green</b> ) hypothesis of the ground-truth structure shown in (a). (d) The Kernel Residual Density profile of the good and bad hypothesis shown in (c). It is expected for a good hypothesis to have high density around its regression surface (equivalently inliers should be densely packed around the regression surface). For a bad hypothesis density would be nearly flat except a very small pear around the regression surface due to small spurious structures. . . . .	50



3.5	<b>Kernel Residual Density Scores</b> of ground truth hypotheses fitted on inliers of the respective structures of <i>johnsona</i> example shown in Fig. 3.1. (a) The raw KRD computed using Eq. 3.2 (b) normalized KRD scores computed using raw KRD in (a), and (c) scaled normalized KRD scores. <i>Note:</i> Here, we used ground truth to demonstrate the motivation behind scaled KRD scores. Throughout, in this work we assume the following are unknowns, (1). <i>number of structures</i> , (2). <i>inliers noise scales</i> , and (3). corresponding <i>number of inliers</i> . . . . .	52
3.6	<b>Residual vs. KRD based point correlation:</b> <i>breadcubechips</i> example of AdelaideRMF [161] dataset (b) Residual based and (c) Density based point correlation (PC) matrix. For better visualization the rows and columns in (b) and (c) are ordered by structure membership. For each of the three structures <i>bread</i> , <i>cube</i> and <i>chips</i> , plots in (d), (e) and (f) show percentage of <i>uncorrelated</i> outliers varying with iterations of (Algo. 1) respectively. . . . .	54
3.7	Termination analysis of Kernel Density Guided Sampling (KDGS). Refer text in Sec. 3.5.8.1 for detail. We show the ground truth inliers in the one of the two views. . . . .	62
3.8	If there are a total of 15 hypotheses in $\mathbf{H}$ , a possible output of Algo. 5, lines 4-7 may look like this. <i>Note:</i> This is just for the illustration purpose, the number of trees and hypotheses may vary. . . . .	67
3.9	Percentage of good hypotheses w.r.t. the outlier rate (%). . . . .	72
3.10	<b>Some Qualitative Results of Motion and Planar Segmentation.</b> Qualitative results are shown for some examples from AdelaideRMF [161] dataset. Motion segmentation: Top-3 rows, Planar Segmentation: Bottom-3 rows. Point membership is color coded. Gross outliers are in red. We have shown only the view 1 of the two views. . . . .	74
3.11	<b>Sample Qualitative Results</b> on York Urban and Toulouse Vanishing Point dataset. . . . .	76
3.12	<b>Multiple Plane Fitting to 3D Point Cloud.</b> Dataset: <i>CastelVecchio</i> and <i>PozzoVeggiani</i> examples from SAMANTHA dataset [43]. Point membership is color coded. . . . .	77
3.13	<b>Multiple Line Fitting.</b> Dataset: <i>Star5</i> [144]. Point membership is color coded. Gross outliers are in red. . . . .	77
3.14	<b>Multiple Circle Fitting.</b> Dataset: <i>Circles5</i> [144]. Point membership is color coded. Gross outliers are in red. . . . .	77
4.1	An adversarial example misclassified by an image Classifier . . . . .	79
4.2	Some adversarial example generation systems. . . . .	79
4.3	Overview of REGroup: <i>Rank-aggregating Ensemble of Generative classifiers for robust predictions</i> . REGroup uses a pre-trained network, and constructs layer-wise generative classifiers modeled by a mixture distribution of the positive and negative pre-activation neural responses at each layer. At test time, an input sample's ( $x$ or $x + \delta$ ) neural responses are tested with generative classifiers to obtain ranking preferences ( $R_+$ and $R_-$ ) of all $m$ classes at all $l$ layers. These preferences are aggregated across all layers using <i>Borda count</i> based preferential voting theory to make final prediction. <i>Note:</i> construction of layer-wise generative classifiers is a one time process. . . . .	83
4.4	cAdv [11] adversarial examples . . . . .	88

4.5	<b>Top-1 and Top-5 accuracy(%) w.r.t PGD adversarial strength.</b> Comparison with adversarial training based method [75] and fine-tuning using random input transformations based method (BaRT) [114] with Expectation Over Transformation (EOT) steps 10 and 40, against the PGD perturbation strength ( $\epsilon$ ). The results of the competing methods are taken from their respective papers. Dataset used: ImageNet-V50K. . . . .	90
4.6	Unrestricted Adversarial Example Via Semantic Manipulation [11]. Row-1 Original Examples, Row-2 Semantically Perturbed Adversarial Examples. .	92
4.7	<b>Physical Attack [162].</b> Stop sign image with adversarial stickers, classified as a speed limit sign. Left: Original stop sign, Middle: Adversarial Mask, Right: Stop Sign classified as speed limit sign. . . . .	93
4.8	Ablation study. Accuracy vs no. of layers ( $k$ ). . . . .	94
4.9	Ablation study. Effect of considering positive and negative pre-activation responses. . . . .	95

# List of Tables

2.1	Depth evaluation result on KITTI Eigen split test set. M: self-supervised monocular supervision, and S: self-supervised stereo supervision, D: depth supervision. '-' means the result is not available from the paper. pRGBD-Refined outperforms all the self-supervised monocular methods and several stereo only and combined monocular and stereo methods. Our results are after 5 <i>self-improving loops</i> . . . . .	30
2.2	Ablation study on 1 <sup>st</sup> self-improving loop. The best performance is in <b>bold</b>	31
2.3	Qualitative depth evaluation on KITTI Odometry sequences 09 and 10. M: self-supervised monocular supervision for fine-tuning. '-' means the result is not available from the paper. Our results are after 5 <i>self-improving loops</i> . Best results in each block is in <b>bold</b> . . . . .	32
2.4	Quantitative depth evaluation results on two TUM <i>freiburg3</i> RGB-D sequences. pRGBD-Refined results are after 3 <i>self-improving loops</i> . . . . .	32
2.5	Quantitative pose evaluation results on KITTI Odometry validation set. '-' means the result is not available from the paper. . . . .	34
2.6	Quantitative pose evaluation results on KITTI Odometry test set. Since the ground truth for the KITTI Odometry test set is not available we run Stereo ORB-SLAM[102] to get the complete camera trajectories and use them as the pseudo ground truth to evaluate. 'X' denotes tracking failure. . . . .	35
2.7	Comparison with state-of-the-art RGB SLAM methods on KITTI Odometry sequences 09 and 10. Here, '-' means the result is not available from the original paper. * denotes the result is obtained from [89]. . . . .	35
2.9	Quantitative pose evaluation results on two TUM <i>freiburg3</i> RGB-D sequences. Note that RGB ORB-SLAM fail in walking_xyz sequence. . . . .	36
2.8	Quantitative pose evaluation results on KITTI Odometry leaderboard. Note that we use the estimated trajectories from ORB-SLAM2-S [102] for global scale alignment. The best performance is in <b>bold</b> . . . . .	37
3.1	<b>Qualitative Evaluation of KDGS (Planar Segmentation)</b> . #H is the total number of hypotheses generated by each method, #HM(%) is the percentage of good hypotheses satisfying the all inlier MSS criteria, #HI(%) percentage of good hypotheses having at-least 80% overlap of their estimated inliers with the true inliers, $n$ is the number of point correspondences, O(%) is the outlier percentage. The $\mathcal{T} = [\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_\kappa]$ vector shows the number of true inliers of all $\kappa$ genuine structures. <i>tim</i> (s) shows the total time taken in seconds. Image shows two-view visual ground-truth inliers with color coded structural membership. Outliers are in red. Best result is in <b>bold</b> and second best is <u>underlined</u> . . . . .	68

3.2 **Qualitative Evaluation of KDGS (Motion Segmentation).** #H is the total number of hypotheses generated by each method, #HM(%) is the percentage of good hypotheses satisfying the all inlier MSS criteria, #HI(%) percentage of good hypotheses having at-least 80% overlap of their estimated inliers with the true inliers,  $n$  is the number of point correspondences, O(%) is the outlier percentage. The  $\mathcal{T} = [\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_\kappa]$  vector shows the number of true inliers of all  $\kappa$  genuine structures.  $tim(s)$  shows the total time taken in seconds. Image shows two-view visual ground-truth inliers with color coded structural membership. Outliers are in red. Best result is in **bold** and second best is underlined. . . . . 69

3.3 **Comparison of KDGS with DHF.** Both DHF and KDGS focus on generating hypothesis for each data point, the plotted hypotheses (in red) are of the inliers of five genuine structures ( $s_1, \dots, s_5$ ) shown in column 1.  $n$  is the total number of data points, O% is the ground-truth (GT) percentage of gross outliers. Ground-truth structure is shown in column 1 (in magenta) and outliers are in green. For better illustration we have plotted hypotheses of each of the five structures separately in the five different rows. . . . . 71

3.4 Required user inputs ( $\epsilon$ = inlier/outlier threshold,  $\kappa$  = no. of structures): ( $\checkmark$  = Required,  $\times$  = Not Required) . In addition to  $\epsilon$ , Prog-X require more user-defined thresholds, details are in [6]. . . . . 71

3.5 **Quantitative Analysis on Motion Segmentation.** Classification Accuracy(CA) in (%), Total time taken including both KDGS and model selection in seconds, O(%)= Outliers Percentage,  $\kappa$  = number of true structures,  $\mu$ =mean,  $med$ =median. DGSAC\* is the DGSAC version proposed in [139]. Results are divided into two block separated by a dashed line. The methods in the top block require either user specified inlier threshold, number of models or both, while methods in second block does not require inlier threshold or number of models. Best result is in bold and second best is underlined. . . . . 73

3.6 **Quantitative Analysis on Planar Segmentation.** Notations are same as of table 3.5. . . . . 75

3.7 **Qualitative Evaluation on VP Estimation.** Notations are same as in table 3.5. . . . . 75

4.1 Dataset used for evaluation and analysis. . . . . 88

4.2 **Comparison with adversarially trained and fine-tuned classification models.** Top-1 and Top-5 classification accuracy (%) of adversarial trained (Inception V3 [75] and ResNet-152 [164]) and fine-tuned (ResNet-50 BaRT [114]) classification models. Clean Images are the non-attacked original images. The results are divided into three blocks, the top block include original networks, middle block include defense approaches based on adversarial re-training/fine-tuning of original networks, bottom block is our defense *without re-training/fine-tuning*. Results of the competing methods are taken from their respective papers. '-' indicate the results were not provided in the respective papers. . . . . 89

4.3 **Performance on Gradient-Based Attacks.** Comparison of Top-1 classification accuracy between SoftMax (SMax) and REGroup based final classification. UN and TA indicates, un-targeted and targeted attacks respectively. The +HC indicates adversarial examples are generated with high-confidence ( $> 90\%$ ) constraint, in this case  $\epsilon$  can be any value that satisfies the HC criteria. For targeted attack we select a target class uniformly at random from the 1000 classes leaving out the true class. #S is the number of images for which the attacker is successfully able to generate adversarial examples using the respective attack models and the accuracies are reported with respect to the #S samples, hence the 0% accuracies with the SoftMax (SMax). Since #S is different for several attacks, therefore, the performance may not be directly comparable *across* different attacks. ‘-’ indicate the information is not-applicable. For Data description refer Tab. 4.1. 91

4.4 **Performance on Physical Adversarial Examples.** Dataset: V10K. Top-1 (%) classification accuracy comparison between SoftMax (SMax) and REGroup. #S is the number of images for which the attacker is successfully able to generate adversarial examples using the respective attack models and the accuracies are reported with respect to the #S samples, hence the 0% accuracies with the SoftMax (SMax). . . . . 93

4.5 **Performance on Gradient-Free Attacks.** Top-1 (%) classification accuracy comparison between SoftMax (SMax) and REGroup. Legends are same as in Tab. 4.3. . . . . 94

4.6 **Inference Time Comparison.** REGroup vs SoftMax: We use a workstation with an i7-8700 CPU and GTX 1080 GPU. . . . . 95



# List of Abbreviations

<b>DGSAC</b>	<b>Density Guided Sampling and Consensus</b>
<b>KRD</b>	<b>Kernel Residual Density</b>
<b>KDGS</b>	<b>Kernel Residual Density Guided Sampling</b>
<b>RANSAC</b>	<b>Random Sampling and Consensus</b>
<b>PMF</b>	<b>Probability Mass Function</b>
<b>MSS</b>	<b>Minimal Sample Subset</b>
<b>GMS</b>	<b>Greedy Model Selection</b>
<b>FHF</b>	<b>Fast Hypothesis Filtering</b>
<b>SLAM</b>	<b>Simultaneous Localization And Mapping</b>
<b>VO</b>	<b>Visual Odometry</b>
<b>pRGBD-SLAM</b>	<b>Pseudo RGB-D SLAM</b>
<b>REGroup</b>	<b>Rank-aggregating Ensemble of Generative Classifiers for Robust Predictions</b>
<b>PGD</b>	<b>Projected Gradient Descent</b>
<b>BPDA</b>	<b>Backward Pass Differentiable Approximation</b>
<b>SPSA</b>	<b>Simultaneous Perturbation, Stochastic Approximation</b>

Dedicated to my parents



# 1 Introduction

## 1.1 Scene Understanding

The ability to understand a scene from its 2D image(s) goes to the heart of many computer vision and learning systems. Here, by *scene*, we mean a space in which an autonomous agent or human can act, interact with, or navigate. Scene understanding can be defined as a process

“to analyze a scene by considering the geometric and semantic context of its contents and the intrinsic relationships between them”[103]

“that reasons jointly about regions, location, class and spatial extent of objects, presence of a class in the image, as well as the scene type”[173]

The ultimate goal of the computer vision is to build an intelligent visual perception system that automatically understands the scene by analyzing the images. This requires the ability to automatically extract the *geometric* and *semantic* information from the raw data (images). Broadly, a scene can be interpreted in the context of its *geometric* structure and that of its *semantic* structure, and both of these contexts can cater to many application scenarios. For example, consider an autonomous driving application; it involves several tasks that rely on the accurate extraction of geometric and semantic information from images. In fig 1.1(a), given an image, we can extract geometric information such as road plane or distance from other vehicles for autonomous visual navigation. We can also extract semantic information such as the Speed limit sign (in Fig. 1.1(c)) or identify road condition in Fig. 1.1(b) (e.g., icy road), to make crucial decisions such as reduce speed, stop the vehicle, or make a turn.

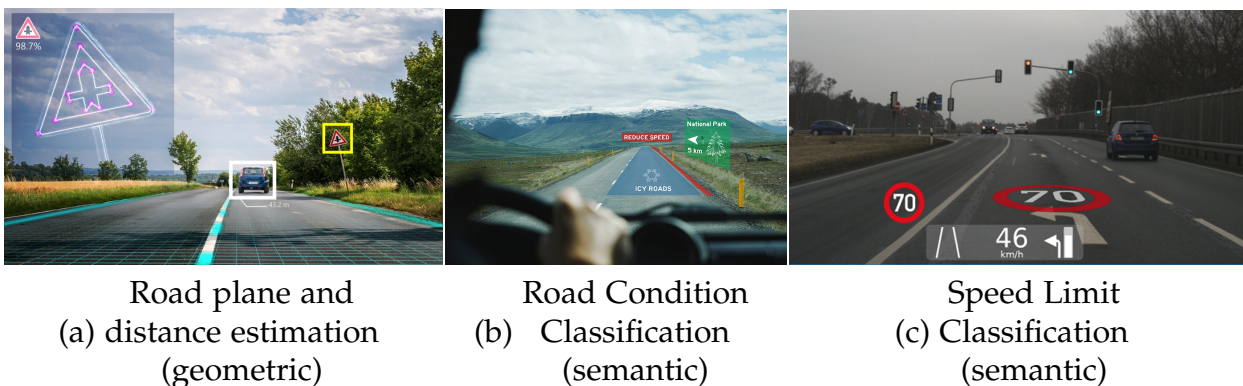


Figure 1.1: Some Autonomous Driving Tasks

Similarly, for Augmented Reality (AR) applications, in order to seamlessly blend the real and digital worlds, it becomes essential to accurately identify the object (*semantic*

information) in the image/video, and accurately estimate its location and pose (*geometric information*) to overlay the augmented content. In Fig. 1.2(a), overlaying the 3D CAD model of a machine part requires correctly identifying the machine part (*semantic information*) and estimate its orientation (*geometric information*) to the cameras, similarly, in Fig. 1.2(b) an AR-based shopping experience requires to identify the items (*semantic information*) and its location with respect to camera (*geometric information*) on the rack to overlay its description alongside it.

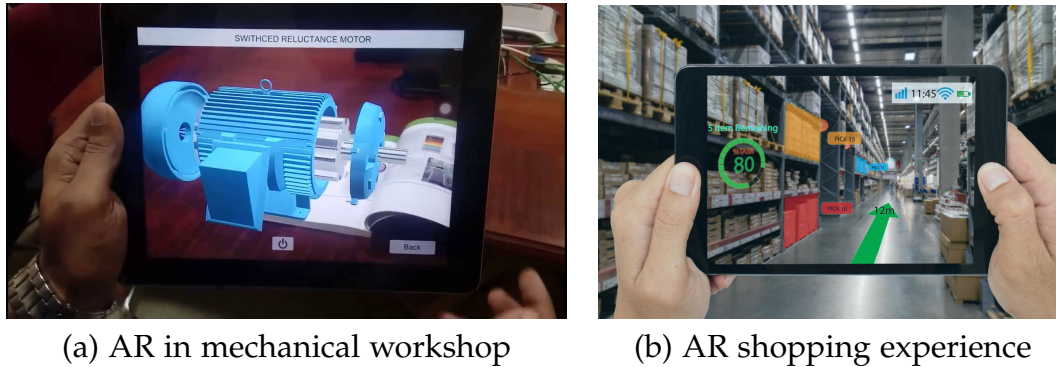


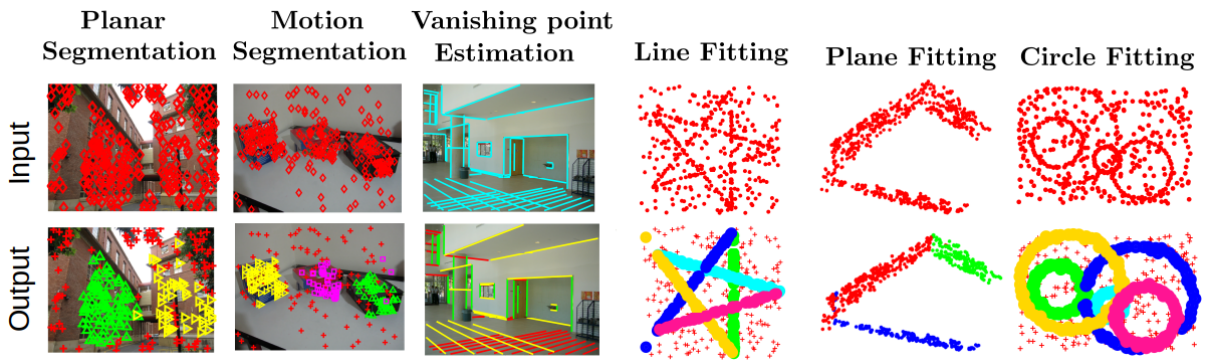
Figure 1.2: Augmented Reality (AR) Applications

**Scene Geometry:** A typical man-made scene (*e.g.* a street) contains many objects like buildings, cars, poles etc., their 3D shapes can be expressed by simple geometric primitives like lines, planes, circles, cylinders in its 2D image. An accurate estimation of such primitives aids in understanding the 3D scene geometry. For example, using the estimated lines, we can further estimate the horizon and the vanishing points, which in turn help estimate the relative orientation of the objects to the ground plane and in the 3D reconstruction of the scene. In Fig. 1.3(a), some of the important geometric scene understanding tasks are shown. Given a pair of images, we can segment the planar regions *e.g.* building facades, road planes (planar segmentation using homography), or motions of objects/persons (motion segmentation using fundamental matrices). From an image pair, we can also find the depth of corresponding scene points (stereo depth estimation). Given a set of RGB images, we can train a learning system to predict per pixel depth from a single image. The predicted depth further can be used to synthesize novel views. Using a monocular video, we can estimate the 3D map of the surrounding environment and simultaneously localize the camera within that environment.

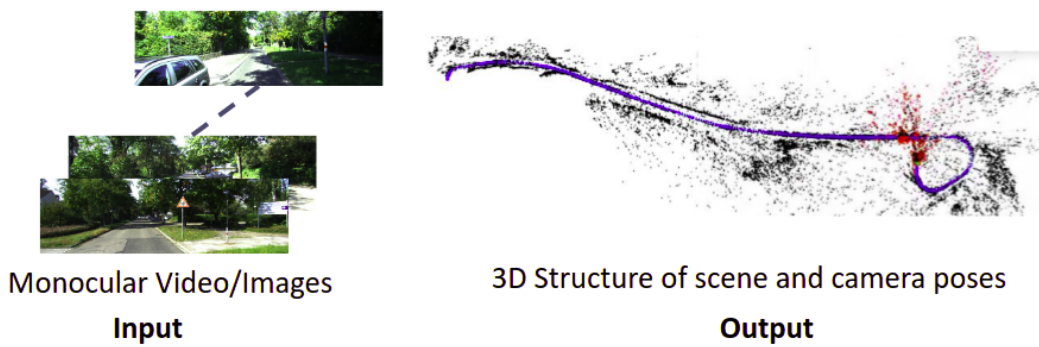
**Scene Semantics:** A scene is a meaningful composition of multiple objects and environments. Semantic information extracted from an image of a scene provide high level information such as whether a particular object is present in the scene (*image classification*), where it is in the image (*object detection*) and the set of pixels corresponding to the object (*semantic segmentation*). This high-level information helps many computer vision applications that require to make decisions automatically *e.g.*, Advanced Driver Assistance System (ADAS), video surveillance, etc. For example, to avoid an accident, it is essential for an ADAS system to correctly classify the road condition, or detect the Speed limit/Stop signs and make decisions such as reduce speed or stop. Object detection and segmentation tasks play an important role in the applications where one needs to localize the object of the interest in the image before further processing. For example, in an AR

application one task could be, detect the object in the image and replace or overlay it with the augmented content. Recent advances in deep learning techniques for object detection and semantic segmentation are closing the gap with respect to human performance. However, they critically depend on the image classification backbone network, which has been shown to be susceptible to adversarial attacks, thus making any downstream task vulnerable. This dependence on Image classification models makes the robustness to adversarial noise and attacks a crucial requirement.

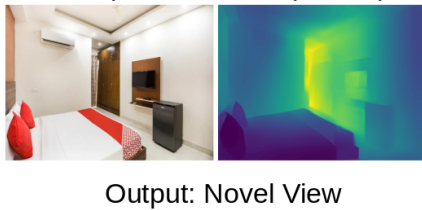
The geometric and semantic scene information forms the basis of many computer vision applications, such as autonomous driving, service/home robotics, video surveillance, that require to automatically make the critical decisions based on this information, Hence, an accurate estimation of both geometric and semantic information becomes essential because inaccurate visual information may lead to a catastrophic failure [138]. These applications rely on several task-specific estimators and predictors (e.g., depth estimator, image classifier, pose estimator) to provide accurate information. To ensure that the extracted information is accurate and is not affected by the unavoidable *task-specific noise* and *outliers* present in the data, it becomes essential to design *robust* scene understanding solutions that can handle task-specific noise and outliers or any adversarial manipulation in the input data.



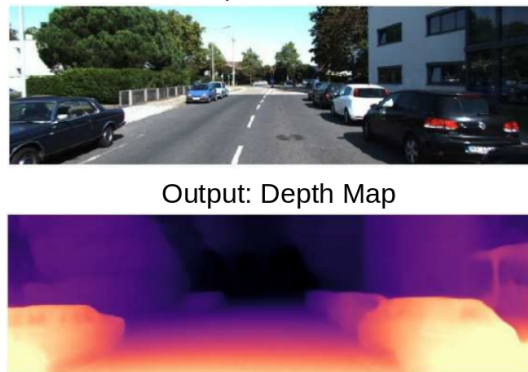
Simultaneous Localization and Mapping (SLAM)



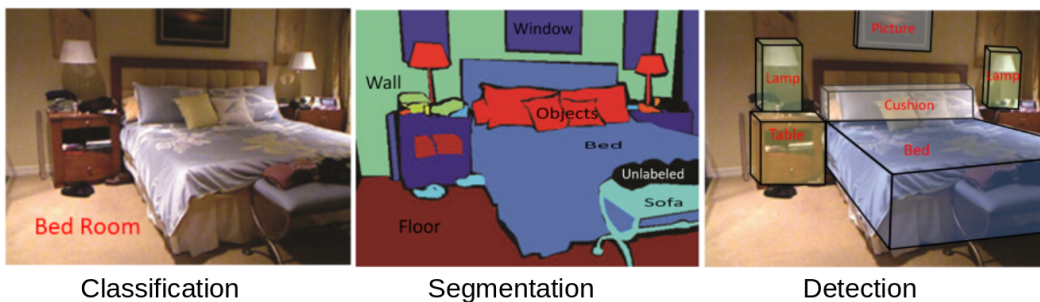
Novel View Synthesis  
Input: RGB + Depth Map



Depth Prediction  
Input: RGB



(a) Illustrative Geometric Scene Understanding Tasks



(b) Illustrative Semantic Scene Understanding Tasks [103]

Figure 1.3: Some Important Scene Understanding Tasks

## 1.2 Towards Robust Scene Understanding

Many automatic decision-making applications like autonomous driving, service, or home robotics rely on both geometric and semantic scene information. Their decisions are influenced by the quality of the geometric and semantic information extracted from the inputs. At the backend of these applications, there are multiple task-specific estimators and predictors that are responsible for providing this information. However, the estimators can make errors, or predictors can give wrong predictions due to noisy inputs, outliers, and violation of modeling assumptions. Here, noise can be sensor noise or artificial noise added by an adversary[132]. The context of noise and outliers varies across different tasks. What is noise or outlier for a task may not be for others. Many deep learning-based solutions are designed under certain assumptions that are often violated under real-world settings, leading them to fail or give poor estimates. For example, the brightness constancy assumption [56], which states a scene point is moving through an image sequence, remains constant. However, this is often violated in real-world settings[126].

What makes human vision unique is its capability to robustly extract meaningful information from the noisy, cluttered, and corrupted visual data. Barrow and Tenenbaum[8] suggest humans perceive information in layers: For example, illumination, Reflectance, Depth, and orientation. To achieve the ultimate goal of an intelligent visual perception system, which is to mimic the human level scene understanding and reasoning from the images and videos, we believe that all the methods intended for scene understanding tasks must consider robustness as one of the necessary components and evaluate their effectiveness for the same.

We next discuss the need for robustness in the context of two geometric and one semantic scene understanding problems addressed in this thesis.

**Geometric Scene Understanding.** *Robust Geometric Model Fitting:* Monocular Simultaneous Localization and Mapping (SLAM) (Fig. 1.4(a)) is one of the important geometric scene understanding tasks, where from a given sequence of images captured from a single camera, the goal is to create a 3D map of the environment and simultaneously localize the camera location with respect to the created 3D map.

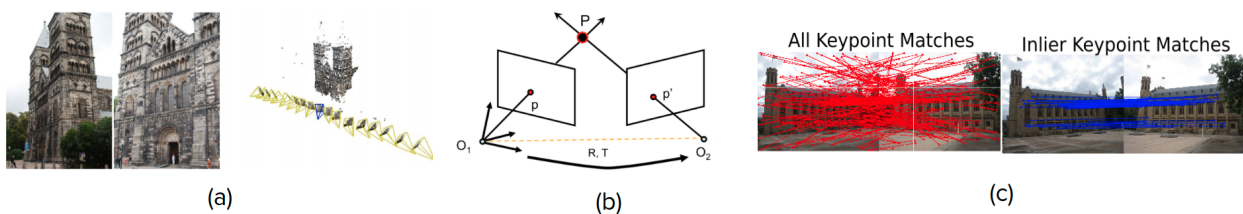


Figure 1.4: (a) Monocular SLAM, (b) Triangulation, (c) Keypoint matches

It consists of three major steps: 3D Map Initialization, tracking, and mapping. Once the 3D map is initialized, the camera location is tracked with respect to the map while simultaneously expanding the 3D map. The map Initialization is a crucial step. A poor initialization leads the SLAM system to fail at the tracking step. In a typical, sparse keypoint-based SLAM, Initialization is done by triangulation process (Fig. 1.4(b)). It

is a process of estimating the 3D location of a scene point using its projected locations (2D keypoints) in the two images captured from two different viewpoints. This process requires the relative pose (motion) between the camera locations of the two images. Generally, a homography or fundamental matrix is estimated from the keypoint matches and decomposed to get the rotation and translation. The processes used to find the keypoint matches are agnostic to the homography or fundamental matrix. Hence they generate many noise and wrong matches (outliers) (Fig. 1.4(c)). In order to automatically correctly estimate the relative pose, a robust mechanism is required that can handle the noisy and outlier keypoint matches without human intervention.

The keypoint-based approach often fails when the scene has low texture. In such scenarios, line features are used in place of key points (Fig. 1.5(a)). In such cases, the relative pose can be estimated using vanishing points and horizon lines [85]. However, similar to keypoint matches methods, the line segment detection methods also output noisy and outlier line segments. Hence, a robust method is required to estimate multiple vanishing points automatically without any user intervention.

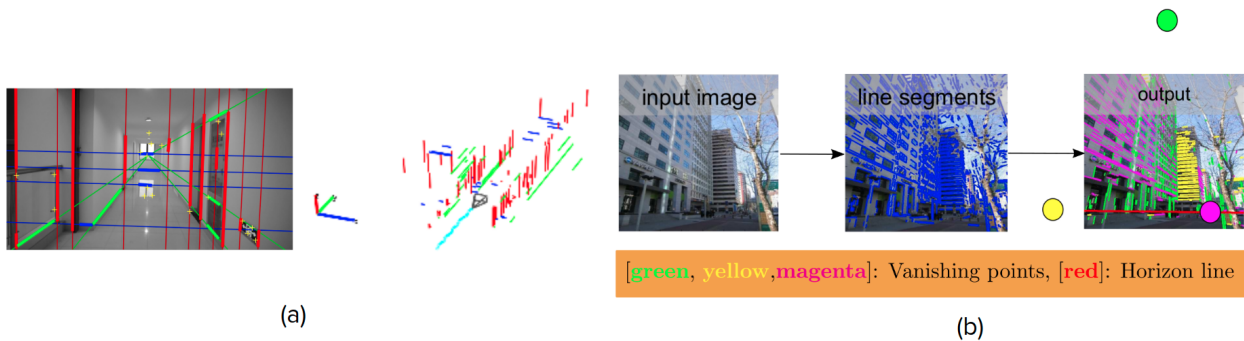


Figure 1.5: (a) Struct SLAM 1.5, (b) Vanishing points and horizon line

Therefore, we propose a generic robust multiple-model fitting framework that can automatically estimate the single and multiple geometric models like vanishing points, fundamental matrix (for motion estimation), homography matrix (for planar surface segmentation), lines, planes, etc.

*Monocular SLAM and Unsupervised CNN-based Depth Prediction:* It is known that monocular RGB-SLAM has well-known limitations in robustness and accuracy compared to those using active depth sensors e.g., RGB-D SLAM [54]. This is due to the inherent scale ambiguity of depth recovery from monocular cameras, which causes the so-called scale drift in both the camera trajectory and 3D scene depth, and thus lowers robustness and accuracy of conventional monocular SLAM. Due to the depth map's availability from the first frame, the 3D map initialization in the RGBD-SLAM is more accurate and robust compared to RGB-SLAM [102].

In most cases, the depth information from an active depth sensor (like Lidar or RGB-D camera) is not available. In such a scenario, we propose to use a CNN-based monocular depth estimation model as a pseudo active depth sensor and run Pseudo RGB-D SLAM. However, unsupervised CNN-based depth models have certain limitations, making them predict un-reliable depth estimates, especially for the distant points. To improve the depth predictions, we propose a self-improving framework to couple an unsupervised CNN-based monocular depth estimation with the geometric SLAM.

Geometric monocular SLAM and unsupervised CNN-based monocular depth prediction represent two largely disjoint approaches towards building a 3D map of the surrounding environment. However, most unsupervised CNN-based monocular depth prediction methods[56, 186] formulate single image depth estimation as a novel view synthesis problem, with appearance-based photometric losses central to their training strategy. Photometric losses primarily rely on the brightness constancy assumption (i.e, neighboring frames don't change much and have consistent brightness). However, this assumption does not always hold, and the performance of these methods degrade where this assumption violates. Another issue with such self-supervised approaches is they operate in a narrow-baseline setting, i.e., they optimize the loss over a short temporal window of 3-5 consecutive frames. Consequently, they fail to predict accurate depth values for distant scene points. An example is shown in Fig. 1.6, where a state-of-the-art monocular depth prediction method called MonoDepth2[56] fails to predict accurate depth estimates for distant scene points corresponding to the van. While it is well known that a wide-baseline yields better depth estimates for points at larger depth, a straightforward extension of existing CNN-based approaches is inadequate for the following two reasons. A wide baseline in a video sequence implies a larger temporal window, which in most practical scenarios will violate the brightness constancy assumption, rendering the photometric loss ineffective. Secondly, larger temporal windows (wider baselines) would also imply more occluded regions that behave as outliers. Unless these aspects are effectively handled, training of CNN-based depth and pose networks in the wide baseline setting will lead to inaccuracies and biases.

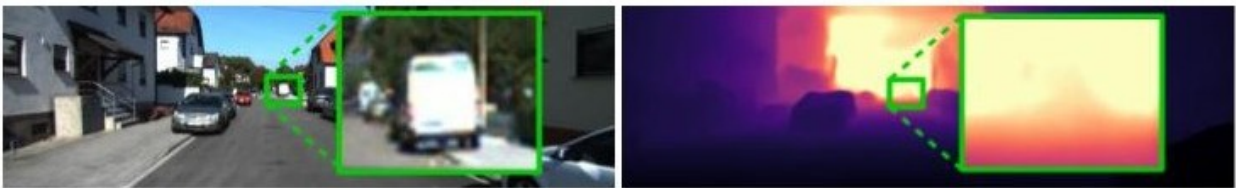


Figure 1.6: *State-of-the-art unsupervised monocular depth prediction method MonoDepth2 [56] fails to predict accurate depth values corresponding to the van.*

Our next contribution is a self-improving, self-supervised geometric-CNN framework. We demonstrate that the coupling of geometric monocular SLAM and unsupervised CNN-based monocular depth prediction by leveraging the strengths of each mitigates the other's shortcomings. While on the one hand, we leverage the depth from CNN to run Pseudo-RGBD SLAM, improving the performance of monocular RGB SLAM, on the other hand, we use the geometric cues from the Pseudo-RGBD SLAM to model the wide baseline constraints and improve the CNN based depth prediction.

In many applications, estimation of these geometric models serve as low-level geometric tasks and are used to fulfill high-level computer vision tasks (e.g., 3D Reconstruction, Obstacle avoidance in autonomous driving). In such scenarios, it becomes essential to design a *generic* multiple geometric model fitting pipeline that can *automatically* extract multiple geometric models from image(s) and can be used as an independent block in different computer vision applications. However, designing such a pipeline requires overcoming several challenges, including noise scale estimation, outlier rejection, and model selection.

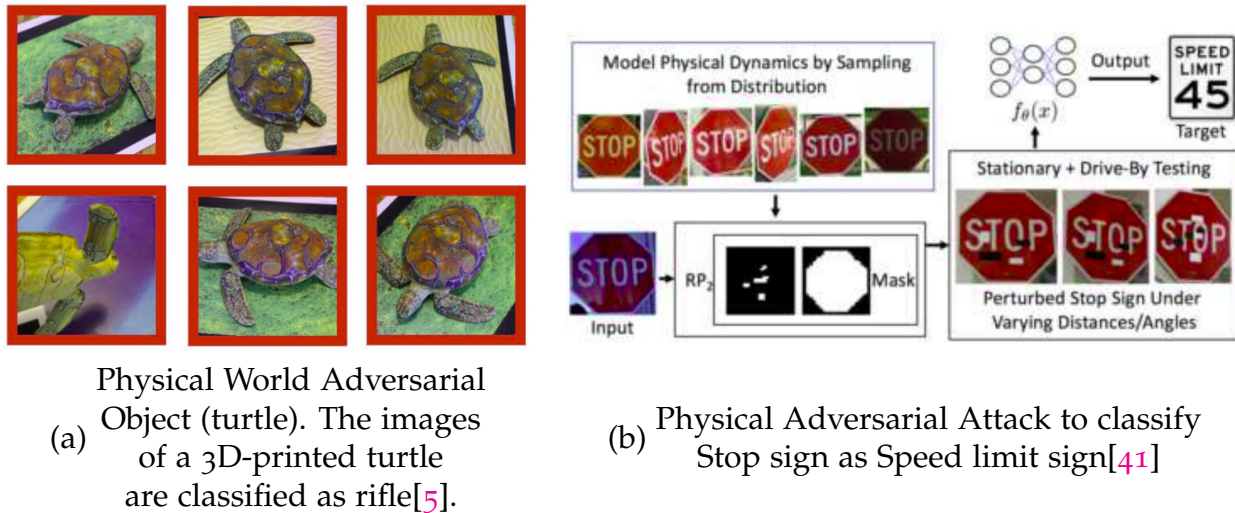


Figure 1.7: Adversarial attack examples, where a classifier fails to predict the correct class

**Semantic Scene Understanding.** *Robust Image Classification:* Recent work has shown that state-of-the-art deep neural networks (DNNs) are not reliable for security-critical applications like biometric systems, autonomous driving, etc. These DNNs are vulnerable to the input samples called adversarial samples (the samples carefully perturbed with the intent of having them misclassified by the DNN [21]). These examples look perceptually very similar to the genuine examples and yet they can easily fool a deep neural network. These adversarial samples are not restricted to digital space (i.e., digital images) but may also be constructed in the physical world, e.g., a deep neural network misclassifies images of a 3D printed turtle to a rifle (Fig. 1.7(a))[5]. An application of a physical-world adversarial attack is autonomous driving, where a stop sign can be modified so as to classify it as a speed-limit sign (Fig. 1.7(b)). Consequently, an autonomous vehicle will increase its speed instead of applying the brake [41]. The adversarial attacks raise the pressing issue of security of the systems supported by the deep neural networks. Critical examples of such systems can be: a physical adversarial example can cause an autonomous vehicle to get crashed [138], or a biometric system can be manipulated to give unlawful, illegal access. The existence of adversarial examples in both digital and physical space raises an essential question on the credibility of deep neural network classifiers in critical security applications.

Classification deep neural networks often form a backbone of detection and segmentation networks. Most popular object detection pipelines [118, 30] involves generating a number of object proposals classifying each of them. Adversarial example generation method[163] generally target proposal block to fool a object detection method. In order to make DNN based image classifier robust to these adversarial attacks, several defense strategies have been proposed. Most of the successful defenses adopt adversarial training or random input transformations that typically require retraining or fine-tuning the model to achieve reasonable performance. A few approaches have been proposed for the large scale image classification task (e.g. using ImageNet [33] level dataset), most of these approaches are based on input transformations or image denoising. Almost all the defenses designed for ImageNet have failed a thorough evaluation. A list of such defenses, along with a thorough evaluation, can be viewed at [90].



Unlike adversarial training and fine-tuning based approaches which are computationally expensive and costly. We propose a robust inference procedure to make a pre-trained image classifier robust to adversarial examples. We statistically analyze the neural responses of intermediate layers of the pre-trained classifier to clean training samples and learn class-specific information in the form of generative classifiers, this is a one time process. During inference, we adopt a preferential voting and rank aggregation approach that combines the evidence(ranked predictions) collected from the intermediate layer generative classifiers and make the final prediction.

## 1.3 Thesis Contributions

In this dissertation, we investigate the robustness for two *geometric* tasks and one *semantic* task towards robust scene understanding. We propose robust solutions for the following three scene understanding tasks.

- (*Geometric*) Self-Supervised, Self-Improving geometric-CNN Framework for 3D Perception
- (*Geometric*) Robust Multiple Geometric Model Fitting
- (*Semantic*) Robust Image Classification

### 1.3.1 Self-Supervised, Self-Improving geometric-CNN Framework for 3D Perception

Given a monocular video or a sequence of monocular images of an scene, we can recover the 3D structure and the orientation of the camera in the environment. The process of simultaneous estimation of the camera orientations and the 3D structure is popularly known as Simultaneous Localization and Mapping (SLAM). A generic feature based SLAM pipeline takes keypoint features as input to estimate the camera orientations and optimize them along with the estimated 3D structure. As we know, features may contain outliers and noise, and may bias the estimated camera orientations and 3D structure. Another method of recovering the 3D map of the environment is to train a deep neural network that takes a single image and predicts the pixel-wise depth map. Both SLAM and depth prediction are two very important geometric scene understanding tasks, that play an important role in application like autonomous driving, visual odometry, visual navigation, novel view synthesis, etc.

#### 1.3.1.1 Our Contributions

Despite the general maturity of monocular geometric SLAM and the rapid advances in unsupervised monocular depth prediction approaches, they both still have their own limitations. Traditional monocular SLAM has well-known limitations in robustness and accuracy as compared to those leveraging active depth sensors, e.g., RGB-D SLAM. This is due to the inherent scale ambiguity of depth recovery from monocular cameras, which causes the so-called scale drift in both the camera trajectory and 3D scene depth, and thus lowers robustness and accuracy of conventional monocular SLAM.

Geometric monocular SLAM and unsupervised CNN based monocular depth prediction represent two largely disjoint approaches towards building a 3D map of the surrounding environment. While on the one hand, geometric RGB-D SLAM can output robust estimate of the camera poses and the sparse 3D feature points, as it apply multiple optimizations over wide baselines. On the other hand, CNN based monocular depth prediction models can provide dense depth maps. We demonstrate that the coupling of these two by leveraging the strengths of each mitigates the other's shortcomings. While it is essential to have wide baseline constraints for accurate depth predictions of the farther away scene points, it is difficult to impose such constraints using photometric error, due to violation of brightness consistency assumption. We propose geometric losses modeling

the wide baseline constraints. Specifically, we propose a joint narrow and wide baseline based self-improving geometric-CNN framework, to improve the performance of both Monocular SLAM and Depth Prediction. A detailed list of contributions is provided in Sec. 2.1.1.

### 1.3.2 Robust Multiple Geometric Model Fitting

Images are 2D projections of a 3D scene, and are rich source of scene’s geometric information. Given an image pair we can extract the 3D structure of the scene, relative camera and object poses, direction of motion of objects, planar surfaces, etc. This information is extracted using low-level features like uniquely identifiable keypoints, line segments, etc., which can be noisy due to scene clutter. The feature extraction techniques are oblivious to the underlying scene geometry, so typically they produce a significant number of outliers, i.e., features that do not adhere to any genuine model instances.

Geometric model fitting plays a vital role in many computer vision applications. Here, a model represents a geometrically meaningful entity of interest. For example, a fundamental matrix in motion segmentation, homography matrix in planar segmentation, Vanishing points in image understanding/3D reconstruction, or geometric primitives like lines, circles, planes, etc.

Many applications require the simultaneous estimation of more than one model instance. Some scenarios where multiple instances can be simultaneously present are shown in Fig.1.8. In Fig. 1.8(a), the motions of multiple objects (here vehicles) are to be estimated by segmenting the keypoint correspondences extracted from an image pair ( Fig. 1.8(a) top row), e.g., the segmented keypoint correspondences are shown in Fig. 1.8(a) bottom row, where red and blue keypoint correspondences correspond to the motion of bus and car respectively. This goal is achieved by estimating the fundamental matrices one for each object motion. The number of fundamental matrices to be estimated is equal to the number of motions. Here, the geometric model is the fundamental matrix that segment the motion via finding the keypoint correspondences that belong to the same object. The above example is one of many such scenarios where the scene changes very frequently. It is unreasonable to expect the user to provide the number of model instances (here, number of moving objects) or fraction of outliers (here, the fraction of keypoint correspondences that do not belong to any genuine motion). Other such scenarios include fitting multiple planes to the 3D point cloud of an indoor scene (Fig. 1.8(b)), or segment coins by fitting circles to each coin (e.g., Fig. 1.8(c)).

We consider model fitting scenarios where the data is generated from multiple model instances (inlier structures) and is corrupted by noise and outliers. Data points that follow a given model are inliers for that model and act as *pseudo-outliers* for all other genuine models, while the ones that do not follow any model are the *gross-outliers*. The goal is to identify all genuine model instances and their corresponding inliers while rejecting the gross outliers. An example of planar surface segmentation is shown in Fig. 1.9. In this example, we find planar surface regions using two images of a scene captured from two different view points. Specifically, we first extract low-level keypoint features (e.g., Scale Invariant Feature Transform (SIFT)[87] ) in the the two-view images and match them across two views to find corresponding keypoints. We extract multiple planar surfaces by fitting multiple planar homographies to the keypoint correspondences in Fig. 1.9(b). The

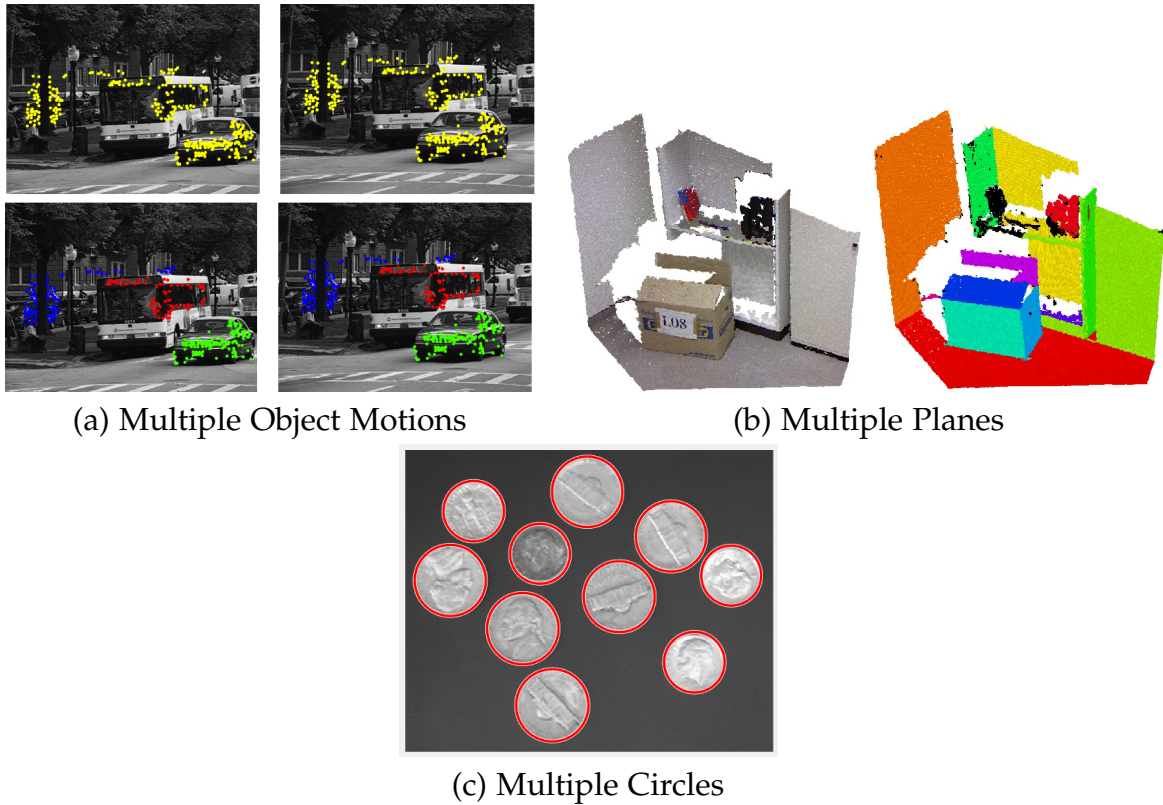


Figure 1.8: Scenarios where multiple instances of the same model can be simultaneously present. (a) Multiple motions, where a motion model of an object is defined by the fundamental matrix estimated using the true keypoint correspondences extracted from an image pair, e.g., fundamental matrix estimated using the red keypoint correspondences defines the motion model of the bus. (b) Fit planes to the point cloud of an indoor scene and (c) Segment coins by fitting circle to each coin.

keypoint correspondences in magenta and blue represent inliers of their respective planar surfaces. The inlier of magenta planar surface act as a pseudo-outliers for planar surface denoted by blue correspondences and vice-versa. The red keypoint correspondences in Fig. 1.9(c) are gross-outliers. The keypoint detection matching algorithms often produce significant number of outliers.

### 1.3.2.1 Our Contributions

A typical multiple model fitting pipeline operating in a hypothesis-and-test framework consists of a hypothesis generation step, followed by model selection using the user-provided inlier threshold and a model selection criteria, and finally point-to-model assignment step. An illustrative example of a multiple model fitting pipeline using a line fitting example is shown in Fig.1.10. In Fig.1.10(b), we fit multiple lines to random subsets of data (also called hypothesis generation). In (c), we select a set of representative hypotheses for each underlying geometric model using a task-specific (here line fitting) inlier threshold and model selection criteria. Finally, in (d), we assign points to each selected model in (c). The users generally provide the number of models and the inlier threshold (any data point whose error to a particular model is less this threshold is referred to as inlier to that particular model).

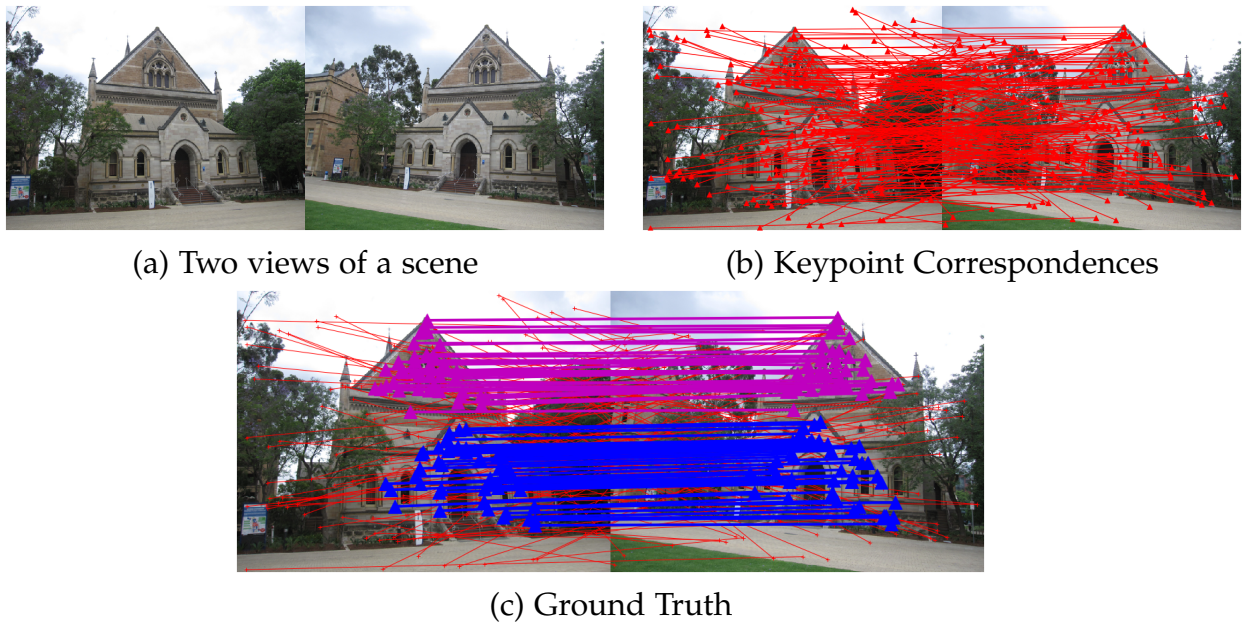


Figure 1.9: *Planar surface segmentation by fitting multiple planar homographies to the keypoint correspondences. In (c), red keypoint correspondences are gross-outliers. The magenta keypoint correspondences act as pseudo-outliers for planar surface denoted by blue keypoint correspondences and vice-versa*

In this work, we automate the various steps in the multiple model fitting pipeline. Specifically, we propose a unified automated multi-model fitting pipeline that can robustly recover multiple geometric models present in a corrupted and noisy data without any user inputs (e.g., the number of models, noise scale, etc.). We use non-parametric density in the space of residual errors and rank ordering based preference analysis as the primary tool to design robust algorithms. The main blocks of the proposed automatic multi model fitting pipeline are guided hypothesis generation, inliers fraction estimation, and model selection. A detailed list of contributions is provided in Sec. 3.1.1.

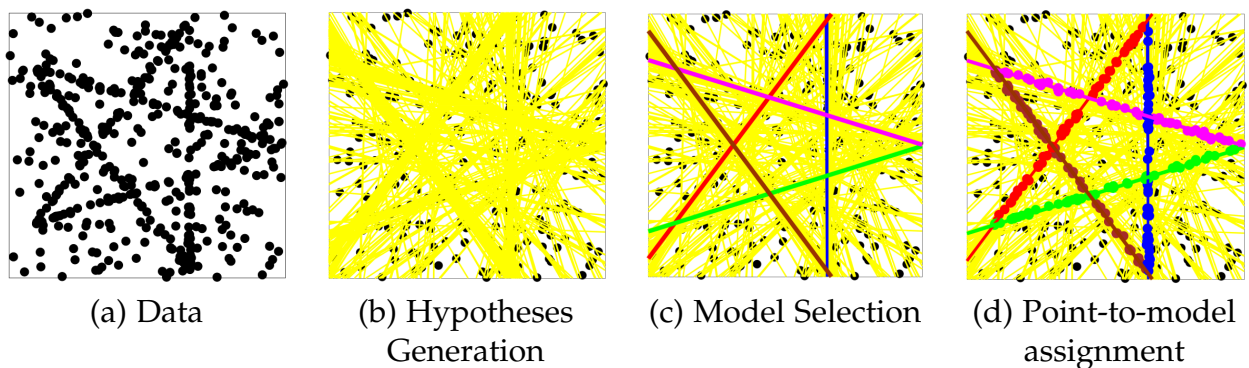


Figure 1.10: *Steps in a typical multiple model fitting pipeline. An illustrative example multiple line fitting.*

### 1.3.3 Robust Image Classification

Image classification is an important task towards semantic scene understanding, which aims to classify an image based on the object present in it. Classification deep neural networks often form a backbone of detection and segmentation networks. The state-of-the-art DL based approaches have achieved human-level accuracy in classifying images [63]. However, recent work has shown that Deep Neural Networks (DNN) based classifiers are not reliable for security-critical applications like biometric systems, autonomous driving, etc. These network models are vulnerable to the input samples called adversarial samples, the samples crafted with the intent of fooling the DNN classifier into misclassifying the input sample.

#### 1.3.3.1 Our Contributions

A pre-trained deep neural network-based image classifier stores the class-specific information in the form of neural network weights. In this work, we statistically analyze the neural responses of intermediate layers of the pre-trained classifier to clean training samples and learn class-specific information in the form of generative classifiers. We adopt a preferential voting and rank aggregation approach that combines the evidence (ranked predictions) collected from the intermediate layer generative classifiers and make the final prediction. The prediction made using our method reduces the adversarial attack success rate by a significant margin (see Sec. 4.6). The proposed method makes a pre-trained network *robust* to adversarial perturbations. Our method, is a *simple, scalable, and practical* defense strategy that is model agnostic and does not require any *re-training* or *fine-tuning*. One can think of our method a test time replacement of SoftMax, which does not compromise the inference time (see Sec. 4.6). A detailed list of contributions is provided in Sec. 4.1.1.

## 1.4 Thesis Organization

In this dissertation, we present robust solutions for the three most important scene understanding tasks towards building a robust visual perception system to bridge the gap between human and computer’s capability to perceive the meaningful information in the presence of noisy and corrupted visual data. We organize the proposed robust solutions chapter wise as follows.

**Self-Improving Monocular SLAM and Depth Prediction Framework** In chapter 2, we formally introduce the problem domain in Sec. 2.1. A discussion on the recent approaches for monocular SLAM and unsupervised depth prediction and their limitations is provided in Sec. 2.2. An overall working and the components of the proposed self-improving framework are introduced in Sec. 2.3. In Sec. 2.3.1 and Sec.2.3.2 we describe pseudo RGB-D for improving monocular SLAM and monocular SLAM for improving depth prediction respectively. The proposed novel training losses are presented in the Sec. 2.3.3. We evaluate the proposed geometric-CNN framework quantitatively and qualitatively in Sec. 2.4 for both monocular SLAM and depth prediction task. In Sec. 2.5, we present a detailed analysis of the self-improving framework, followed by a discussion in Sec. 2.6.

**A Generic Robust Multiple Geometric Model Fitting Pipeline.** In chapter 3, we formally introduce the problem domain in Sec. 3.1. A categorical discussion on the recent approaches for guided sampling and full multi-model fitting and their limitations is provided in Sec. 3.2.1, a formal problem statement is described in Sec. 3.3 followed by the overall methodology of the proposed pipeline in Sec. 3.4. The components of the proposed unified, robust automated multiple model fitting pipeline are introduced in Sec. 3.5.2 (Kernel Residual Density) , in Sec. 3.5.5 (guided hypotheses generation), in Sec. 3.5.9 (inlier noise scale estimation) and model selection algorithms in Sec. 3.5.10. The extensive experimental results analysis on wide variety of model fitting tasks like planar segmentation, motion segmentation, line fitting, plane fitting, circle fitting, and vanishing point estimation is shown in Sec. 3.6.

**Robust Image Classification.** In chapter 4, we first introduce the problem domain in Sec. 4.1. We discuss the defense approaches to adversarial attacks in Sec. 4.2. We specifically discuss defenses for large scale image classification tasks in Sec. 4.2.3. An overall methodology of the proposed defense strategy is presented in Sec. 4.3. In Sec. 4.4, we discuss the process of extracting class-specific information from the intermediate layers of a neural network and building class-specific generative classifiers. In Sec. 4.5 we present a rank aggregation and preferential voting based mechanism to combine the evidences from the intermediate layers for final predictions. We discuss various attack generation methods and present detailed experimental analysis in Sec. 4.6. Further analysis of gradient masking and ablation study is presented in Sec. 4.6.5 and Sec. 4.6.6 respectively. A detailed discussion is presented in Sec. 4.7.

In chapter 5, we present the thesis conclusion by summarizing the contributions and propose several perspectives about future research directions.

## 1.5 Outcomes of this Thesis

### 1.5.1 Publications

#### Conferences

- **Lokender Tiwari**, Pan Ji, Quoc-Huy Tran, Bingbing Zhuang, Saket Anand, and Manmohan Chandraker. "*Pseudo RGB-D for Self-Improving Monocular SLAM and Depth Prediction*", European Conference on Computer Vision (ECCV), 2020. [arXiv Version \[143\]](#)
- **Lokender Tiwari**, Saket Anand. "*DGSAC: Density Guided SAMpling and Consensus*", IEEE Winter Conference on Applications of Computer Vision (WACV), 2018, pp. 974-982. [Author Version \[139\]](#)
- **Lokender Tiwari**, Saket Anand, and Sushil Mittal. "*Robust multi-model fitting using density and preference analysis.*" Asian Conference on Computer Vision (ACCV), 2016, pp. 308-323. Springer. [Author Version\[141\]](#)
- **Lokender Tiwari**, Saket Anand. "*Fast hypothesis filtering for multi-structure geometric model fitting.*" IEEE International Conference on Image Processing (ICIP). 2016, pp. 3728-3732. [Author Version \[140\]](#)

#### Technical Reports

- **Lokender Tiwari**, Anish Madan, Saket Anand, Subhashis Banerjee, "*Dissecting Deep Networks into an Ensemble of Generative Classifiers for Robust Predictions*", arXiv, 2020. [arXiv Version \[142\]](#)
- **Lokender Tiwari**, Saket Anand, "*DGSAC: Density Guided Sampling and Consensus: A Unified Framework for Automatic Robust Multiple Structure Recovery*".
- **Lokender Tiwari**, Anish Madan, Saket Anand, Subhashis Banerjee, "*REGroup: Rank Aggregating Ensemble of Generative Classifiers for Robust Predictions*".

### 1.5.2 Patent Applications

Quoc-Huy Tran, Lokender Tiwari, Pan Ji, Manmohan Chandraker  
*Pseudo-RGB-D: Exploiting Learned Depth for Robust SfM.*  
 NEC Laboratories America, Inc.  
 (Application #US/16/987,705, filed on Aug 2020)

### 1.5.3 Awards

- [IEEE WACV 2018 PhD Forum Award](#), Mentor Prof. Terrance Boulton
- [Best Doctoral Symposium Award \(3rd\) Tenth Indian Conference on Computer Vision, Graphics and Image Processing, 2016](#)
- [IEEE Signal Processing Society Travel Award 2016](#)
- [Visvesvarya PhD Fellowship Award](#)



- Best Poster (2nd), Research Showcase 2017 @ [IIIT-Delhi](#)

#### 1.5.4 Open Source Code and Project Pages

- Density Guided Sampling and Consensus (DGSAC) [Github](#)
- Self-Improving geometric-CNN Framework [Project Page](#), [Demos](#)
- Robust Image Classification [Project Page](#)
- Density Preference Analysis [Github](#)



## 2 Self-Improving geometric-CNN Framework for 3D Perception

*“There are things known and there are things unknown, and in between are the doors of perception”*

– Aldous Huxley (1894 - 1963)

### 2.1 Introduction

One of the most reliable cues towards 3D perception from a monocular camera arises from camera motion that induces multiple-view geometric constraints [62] wherein the 3D scene structure is encoded. Over the years, Simultaneous Localization and Mapping (SLAM) [32, 70, 104] has been long studied to simultaneously recover the 3D scene structure of the surrounding and estimate the ego-motion of the agent. With the advent of Convolutional Neural Networks (CNNs), unsupervised learning of single-view depth estimation [51, 55, 186] has emerged as a promising alternative to the traditional geometric approaches. Such methods rely on CNNs to extract meaningful depth cues (e.g., shading, texture, and semantics) from a single image, yielding very promising results.

Despite the general maturity of monocular geometric SLAM [38, 101, 37] and the rapid advances in unsupervised monocular depth prediction approaches [96, 150, 176, 12, 56, 127], they both still have their own limitations.

**Monocular RGB SLAM.** Traditional monocular SLAM has well-known limitations in robustness and accuracy as compared to those leveraging active depth sensors, e.g., RGB-D SLAM [102]. For example, if we run a popular, widely used RGB ORB-SLAM2 [102] on sequence 01 of KITTI Odometry dataset it fails after some time, while if we run RGB-D ORB-SLAM2 [102], it succeeds. However, an important point here to note is, to run RGB-D SLAM, the depth (D) is required from an active depth sensor e.g., LiDAR. In most cases, we do not have access to an active depth sensor. This performance issue is due to the inherent scale ambiguity of depth recovery from monocular cameras, which causes the so-called scale drift in both the camera trajectory and 3D scene depth, and thus lowers robustness and accuracy of conventional monocular SLAM. In addition, the triangulation-based depth estimation employed by traditional SLAM methods is degenerate under pure rotational camera motion [62].

**Unsupervised Monocular Depth Prediction.** Most of the unsupervised and self-supervised depth estimation methods [186, 55, 56, 12] formulate single image depth estimation as a novel-view synthesis problem, with appearance-based photometric losses being central to the training strategy. Usually, these models train two networks, one each for *pose* and *depth*. As photometric losses largely rely on the brightness constancy assumption, nearly

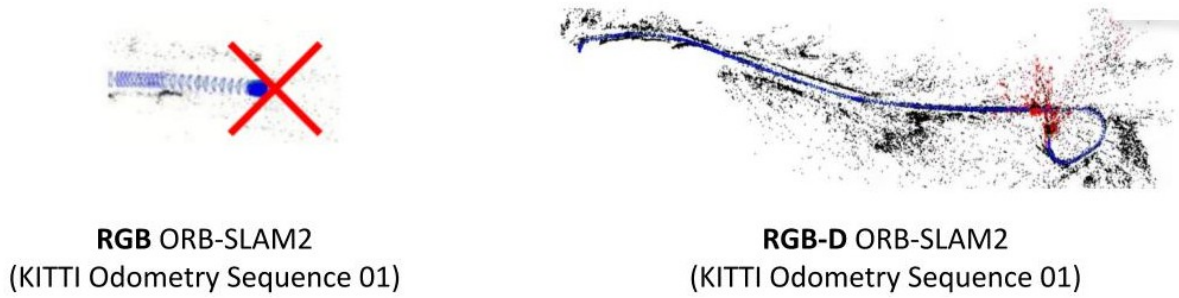


Figure 2.1: RGB SLAM vs. RGB-D SLAM Robustness and Accuracy. RGB ORB-SLAM2 [102] fails to track after sometime, on KITTI benchmark dataset [52] sequence 01, RGB-D ORB-SLAM2 [102] successfully build the 3D map without tracking failure.

all existing self-supervised approaches operate in a narrow-baseline setting, optimizing the loss over a snippet of 2-5 consecutive frames. Typically, a novel view synthesis based depth prediction model (e.g., MonoDepth2) takes three consecutive frames as input. The predicted depth of the central frame and the predicted relative poses of the neighboring frames is used to synthesize the neighboring frames. The photometric reconstruction error computed on the synthesized frames with the original frames is used to train the depth model. Since the depth of the central frame is conditioned on the narrow-baseline (i.e., synthesis of the just neighboring frames), depth models tend to predict depth estimates of the distant points with high uncertainty. For example, in Fig. 2.2, the depth estimates for points that are farther away corresponding to VAN and Sign pole are inaccurate. While it is well known that a wide-baseline yield better depth estimates for points at larger depth, a straightforward extension of existing CNN based approaches is inadequate for the following two reasons. A wide baseline in a video sequence implies a larger temporal window, which in most practical scenarios, will violate the brightness constancy assumption, rendering the photometric loss ineffective. Secondly, larger temporal windows (wider baselines) would also imply more occluded regions that behave as outliers. Unless these aspects are effectively handled, training of CNN based depth and pose networks in the wide baseline setting will lead to inaccuracies and biases.

In view of the limitations in both monocular geometric SLAM and unsupervised monocular depth estimation approaches, a particularly interesting question to ask is whether these two approaches can complement each other (see Sec. 2.5) and mitigate the issues discussed above. Our work makes contributions towards answering this question. Specifically, we propose a *self-supervised, self-improving* framework of these two tasks, which is shown to improve the robustness and accuracy of each of them.

While the performance gap between geometric SLAM and self-supervised learning-based SLAM methods is still large, incorporating depth information drastically improves the robustness of geometric SLAM methods (e.g., see RGB-D SLAM vs. RGB SLAM on the KITTI Odometry leaderboard [52]). Inspired by this success of RGB-D SLAM, we postulate the use of an unsupervised CNN-based depth estimation model as a *pseudo depth sensor*, which allows us to design our self-supervised approach, pseudo RGB-D SLAM (pRGBD-SLAM) that only uses monocular cameras and yet achieves significant improvements in robustness and accuracy as compared to RGB SLAM.



Figure 2.2: State-of-the-art unsupervised monocular depth prediction method MonoDepth2 [56] fails to predict accurate depth values for both the farther away points (top 2 rows) and also nearby points (bottom row).

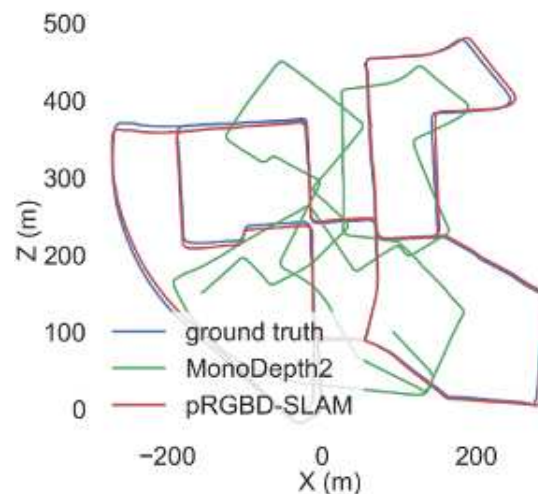


Figure 2.3: MonoDepth2 [56] pose network camera poses vs. pseudo RGBD-SLAM camera poses where depth( $D$ ) is from CNN. The pose network from [56] leads to significant drift.

Our fusion of geometric SLAM and CNN-based monocular depth estimation turns out to be symbiotic, and this complementary nature sets the basis of our self-improving framework. To improve the depth predictions, we make use of two main modifications in the training strategy. First, we eschew the learning-based pose estimates in favor of geometric SLAM based estimates (an illustrative motivation is shown in Fig. 2.3). Second, we make use of common tracked keypoints from neighboring *keyframes* and impose a symmetric depth transfer and a depth consistency loss on the CNN model. These adaptations are based on the observation that both pose estimates and sparse 3D feature point estimates from geometric SLAM are robust, as most techniques typically apply multiple bundle adjustment iterations over wide baseline depth estimates of

common keypoints. This simple observation and the subsequent modification is key to our self-improving framework, which can leverage any unsupervised CNN-based depth estimation model and a modern monocular SLAM method. In this work, we test our framework, with ORBSLAM [102] as the geometric SLAM method and MonoDepth2 [56] as the CNN-based model. We show that our self-improving framework outperforms previously proposed self-supervised approaches that utilize monocular, stereo, and monocular-plus-stereo cues for self-supervision (see Tab. 2.1) and a strong feature-based RGB-SLAM baseline (see Tab. 2.6).

In this work, we demonstrate that the coupling of these two by leveraging the strengths of each mitigates the other’s shortcomings. Specifically, we propose a joint narrow and wide baseline based self-improving framework, where on the one hand, the CNN-predicted depth is leveraged to perform *pseudo RGB-D* feature-based SLAM, leading to better accuracy and robustness than the monocular RGB SLAM baseline. On the other hand, the bundle-adjusted 3D scene structures and camera poses from the more principled geometric SLAM are injected back into the depth network through novel wide baseline losses proposed for improving the depth prediction network, which then continues to contribute towards better pose and 3D structure estimation in the next iteration. We emphasize that our geometry-CNN framework only requires *unlabeled monocular* videos in both training and inference stages, and yet is able to outperform state-of-the-art *self-supervised monocular* and *stereo* depth prediction networks (e.g., Monodepth2) and feature-based monocular SLAM system (i.e., ORB-SLAM).

The framework runs in a simple alternating update fashion: first, we use depth maps from the CNN-based depth network and run pRGBD-SLAM; second, we inject the outputs of pRGBD-SLAM, i.e., the relative camera poses and common tracked keypoints and keyframes to fine-tune the depth network parameters to improve the depth prediction; then, we repeat the process until we see no improvement.

### 2.1.1 Contributions

Our specific contributions are summarized here:

- We propose a self-improving strategy to inject into depth prediction networks the supervision from SLAM outputs, which stem from more generally applicable geometric principles.
- We introduce two wide baseline losses, i.e., the symmetric depth transfer loss and the depth consistency loss on common tracked points, and propose a joint narrow and wide baseline based depth prediction learning setup, where appearance-based losses are computed on narrow baselines and purely geometric losses on wide baselines (non-consecutive temporally distant keyframes).
- Through extensive experiments on KITTI [52] and TUM RGB-D [131], our framework is shown to outperform both monocular SLAM system (i.e., ORB-SLAM [101]) and the state-of-the-art unsupervised single-view depth prediction network (i.e., Monodepth2 [56]).

The remainder of the chapter is organized as follows. We categorically discuss recent approaches for monocular SLAM and unsupervised depth prediction, and their limitations is provided in Sec. 2.2. An overall working and the components of the proposed self-improving framework are introduced in Sec. 2.3. In Sec. 2.3.1 and Sec.2.3.2 we describe pseudo RGB-D for improving monocular SLAM and monocular SLAM for improving depth prediction respectively. The novel losses are presented in the Sec. 2.3.3. We evaluate the proposed self-improving geometric-CNN framework quantitatively and qualitatively in Sec. 2.4 for both monocular SLAM and depth prediction task. In Sec. 2.5, we present a detailed analysis of the self-improving framework, followed by a discussion in Sec. 2.6.

## 2.2 Monocular SLAM and Depth Prediction Approaches

### 2.2.1 Monocular SLAM Approaches

Visual SLAM has a long history of research in the computer vision community. Due to its well-understood underlying geometry, various geometric approaches have been proposed in the literature, ranging from the classical MonoSLAM [32], PTAM [70], DTAM [104] to the more recent LSD-SLAM [38], ORB-SLAM [101] and DSO [37]. More recently, in view of the successful application of deep learning in a wide variety of areas, researchers have also started to exploit deep learning approaches for SLAM, in the hope that it can improve certain components of geometric approaches or even serve as a complete alternative. Our work makes further contributions along this line of research.

### 2.2.2 Unsupervised Monocular Depth Prediction Approaches

Inspired by the pioneering work by Eigen et al. [36] on learning single-view depth estimation, a vast amount of learning methods emerge along this line of research. The earlier works often require ground truth depths for fully-supervised training. However, per-pixel depth ground truth is generally hard or prohibitively costly to obtain. Therefore, many self-supervised methods that make use of geometric constraints as supervision signals are proposed. Specifically, thanks to the Spatial Transform Network [65], differentiable photometric reconstruction loss is successfully applied to monocular depth estimation. One example work is by Godard et al. [55], which relies on the photo-consistency between the left-right cameras of a calibrated stereo. Zhou et al. [186] go one step further to learn monocular depth prediction as well as ego-motion estimation, thereby permitting unsupervised learning with only a monocular camera. This pipeline has inspired a large amount of follow-up works that utilize various additional heuristics, including 3D geometric constraints on point clouds [96], direct visual odometry [150], joint learning with optical flow [176], scale consistency [12], and others [56, 127].

### 2.2.3 Approaches Using Depth to Improve Monocular SLAM

Approaches [133, 175, 170, 86] leveraging CNN-based depth estimates to tackle issues in monocular SLAM have been proposed. CNN-SLAM [133] uses learned depth maps to initialize keyframes' depth maps in LSD-SLAM [38] and refines them via a filtering framework. Yin et al. [175] use a combination of CNNs and conditional random fields

to recover scale from the depth predictions and iteratively refine ego-motion and depth estimates.

Recently, DVSO [170] trains a single CNN to predict both the left and right disparity maps, forming a virtual stereo pair. The CNN is trained with photo-consistency between stereo images and consistency with depths estimated by Stereo DSO [156]. More recently, CNN-SVO [86] uses depths learned from stereo images to initialize depths of keypoints and reduce their corresponding uncertainties in SVO [46]. In contrast to our self-supervised approach, [133, 175] use *ground truth* depths for training depth networks while [170, 86] need *stereo* images.

#### 2.2.4 Approaches Using SLAM to Improve Monocular Depth Prediction

Depth estimates from geometric SLAM have been leveraged for training monocular depth estimation networks in recent works [71, 1]. In [1], sparse depth maps by Stereo ORB-SLAM [102] are first converted into dense ones via an auto-encoder, which are then integrated into geometric constraints for training the depth network. Klodt and Vedaldi [71] employ depths and poses by ORB-SLAM [101] as supervision signals for training the depth and pose networks respectively. This approach only considers five consecutive frames, thus restricting its operation in the narrow-baseline setting.

### 2.3 Self-Improving geometric-CNN Framework

Our self-improving framework leverages the strengths of each, the unsupervised single-image depth estimation and the geometric SLAM approaches, to mitigate the other’s shortcomings. On one hand, the depth network typically generates reliable depth estimates for nearby points, which assist in improving the geometric SLAM estimates of poses and sparse 3D points (Sec. 2.3.1). On the other hand, geometric SLAM methods rely on a more holistic view of the scene to generate robust pose estimates as well as identify *persistent* 3D points that are visible across many frames, thus providing an opportunity to perform wide-baseline and reliable sparse depth estimation. Our framework leverages these sparse, but robust estimates to improve the noisier depth estimates of the farther scene points by minimizing a blend of the symmetric transfer and depth consistency losses (Sec. 2.3.2) and the commonly used appearance based loss. In the following iteration, this improved depth estimate further enhances the capability of geometric SLAM and the cycle continues until the improvements become negligible. Even in the absence of ground truth, our self-improving framework continues to produce better pose and depth estimates.

An overview of the proposed self-improving framework is shown in Fig. 2.4, which iterates between improving poses and improving depths. Our pose refinement and depth refinement steps are then detailed in Sec. 2.3.1 and 2.3.2 respectively. An overview of narrow and wide baseline losses we use for improving the depth network is shown in Fig. 2.5 and details are provided in Sec. 2.3.2.



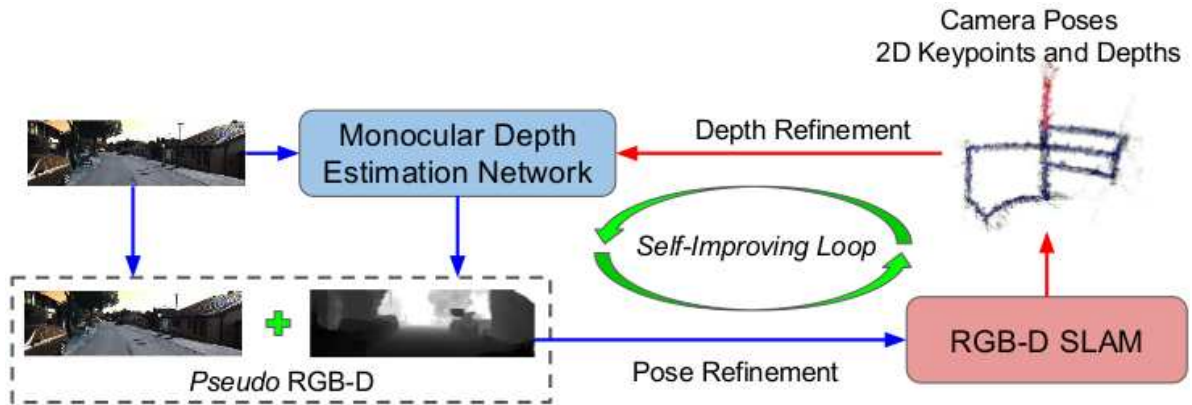


Figure 2.4: Overview of Our Self-Improving Framework. It alternates between pose refinement (blue arrows; Sec. 2.3.1) and depth refinement (red arrows; Sec. 2.3.2).

### 2.3.1 Pseudo RGB-D for Improving Monocular SLAM

**Pose Estimation / Refinement.** We employ a well explored and widely used geometry-based SLAM system, i.e., the RGB-D version of ORB-SLAM [102], to process the pseudo RGB-D data, yielding camera poses as well as 3D map points and the associated 2D keypoints. Any other geometric SLAM system that provides these output estimates can also be used in place of ORB-SLAM. A trivial direct use of pseudo RGB-D data to run RGB-D ORB-SLAM is not possible, because CNN might predict depth at a very different scale compared to depth measurements from real active sensors, e.g., LiDAR. Keeping the above difference in mind, we discuss an important adaptation in order for RGB-D ORB-SLAM to work well in our setting. We first note that RGB-D ORB-SLAM transforms the depth data into disparity on a virtual stereo to reuse the framework of stereo ORB-SLAM. Specifically, considering a keypoint with 2D coordinates  $(u_l, v_l)$  (i.e.,  $u_l$  and  $v_l$  denote the horizontal and vertical coordinates respectively) and a CNN-predicted depth  $d_l$ , the corresponding 2D keypoint coordinates  $(u_r, v_r)$  on the virtual rectified right view are  $u_r = u_l - \frac{f_x b}{d_l}$ ,  $v_r = v_l$ , where  $f_x$  is the horizontal focal length and  $b$  is the virtual stereo baseline.

**Adaptation.** In order to have a reasonable range of disparity, we mimic the setup of the KITTI dataset [52] by making the baseline adaptive,  $b = \frac{b^{\text{KITTI}}}{d_{\text{max}}^{\text{KITTI}}} * d_{\text{max}}$ , where  $d_{\text{max}}$  represents the maximum CNN-predicted depth of the input sequence, and  $b^{\text{KITTI}} = 0.54$  and  $d_{\text{max}}^{\text{KITTI}} = 80$  (both in meters) are respectively the actual stereo baseline and empirical maximum depth value of the KITTI dataset.

We also summarize the overall pipeline of RGB-D ORB-SLAM here. The 3D map is initialized at the very first frame of the sequence due to the availability of depth. After that, the following main tasks are performed: i) track the camera by matching 2D keypoints against the local map, ii) enhance the local map via local bundle adjustment, and iii) detect and close loops for pose-graph optimization and full bundle adjustment to improve camera poses and scene depths. As we will show in Sec. 2.4.4, using pseudo RGB-D data leads to better robustness and accuracy as compared to using only RGB data.

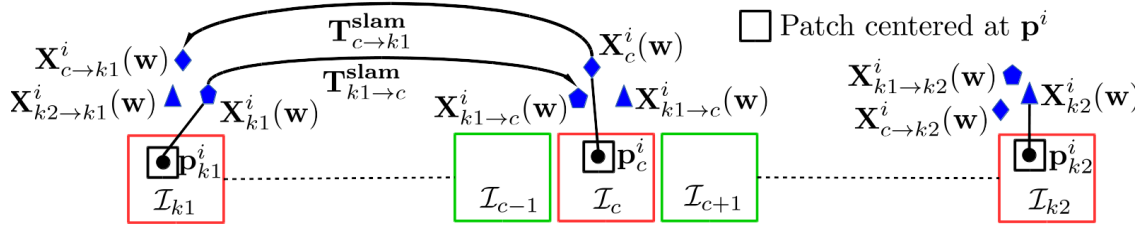


Figure 2.5: *Narrow and Wide Baseline Losses. Narrow baseline photometric and smoothness losses involve keyframe  $\mathcal{I}_c$  and temporally adjacent frames  $\mathcal{I}_{c-1}$  and  $\mathcal{I}_{c+1}$ , and wide baseline symmetric depth transfer and depth consistency losses involve keyframe  $\mathcal{I}_c$  and temporally farther keyframes  $\mathcal{I}_{k1}$  and  $\mathcal{I}_{k2}$ . Refer to the text below for details.*

### 2.3.2 Monocular SLAM for Improving Depth Prediction

Depth Prediction / Refinement. We start from the pre-trained depth network of Monodepth2 [56], a state-of-the-art monocular depth estimation network, and fine-tune its network parameters with the camera poses, 3D map points and the associated 2D keypoints produced by the above pseudo RGB-D ORB-SLAM (pRGBD-SLAM). In contrast to Monodepth2, which relies only on the narrow baseline photometric reconstruction loss between adjacent frames for short-term consistencies, we propose wide baseline symmetric depth transfer and sparse depth consistency losses to introduce long-term consistencies. Our final loss (Eq. (2.4)) consists of both narrow and wide baseline losses. The narrow baseline losses, *i.e.*, photometric and smoothness losses, involve the current keyframe  $\mathcal{I}_c$  and its temporally adjacent frames  $\mathcal{I}_{c-1}$  and  $\mathcal{I}_{c+1}$ , while wide baseline losses are computed on the current keyframe  $\mathcal{I}_c$  and the two neighboring keyframes  $\mathcal{I}_{k1}$  and  $\mathcal{I}_{k2}$  that are temporally farther than  $\mathcal{I}_{c-1}$  and  $\mathcal{I}_{c+1}$  (see Fig. 2.5). Next, we introduce the notation and describe the losses in detail.

#### 2.3.3 Narrow and Wide Baseline Losses

**Notation.** Let  $\mathcal{X}$  represent the set of common tracked keypoints visible in all the three keyframes  $\mathcal{I}_{k1}$ ,  $\mathcal{I}_c$  and  $\mathcal{I}_{k2}$  obtained from pRGBD-SLAM. Note that  $k1$  and  $k2$  are two neighboring keyframes of the current frame  $c$  (*i.e.*,  $k1 < c < k2$ ) in which keypoints are visible. Let  $\mathbf{p}_{k1}^i = [p_{k1}^{i1}, p_{k1}^{i2}]$ ,  $\mathbf{p}_c^i = [p_c^{i1}, p_c^{i2}]$  and  $\mathbf{p}_{k2}^i = [p_{k2}^{i1}, p_{k2}^{i2}]$  be the 2D coordinates of the  $i^{th}$  common tracked keypoint in the keyframes  $\mathcal{I}_{k1}$ ,  $\mathcal{I}_c$  and  $\mathcal{I}_{k2}$  respectively, and the associated depth values obtained from pRGBD-SLAM are represented by  $d_{k1}^i(\text{SLAM})$ ,  $d_c^i(\text{SLAM})$ , and  $d_{k2}^i(\text{SLAM})$  respectively. The depth values corresponding to the keypoints  $\mathbf{p}_{k1}^i$ ,  $\mathbf{p}_c^i$  and  $\mathbf{p}_{k2}^i$  can also be obtained from the depth network and are represented by  $d_{k1}^i(\mathbf{w})$ ,  $d_c^i(\mathbf{w})$ , and  $d_{k2}^i(\mathbf{w})$  respectively, where  $\mathbf{w}$  stands for the depth network parameters.

##### 2.3.3.1 Symmetric Depth Transfer Loss

Given the camera intrinsic matrix  $\mathbf{K}$ , and the depth value  $d_c^i(\mathbf{w})$  of the  $i^{th}$  keypoint  $\mathbf{p}_c^i$ , the 2D coordinates of the keypoint  $\mathbf{p}_c^i$  can be back-projected to its corresponding 3D coordinates as:  $\mathbf{X}_c^i(\mathbf{w}) = \mathbf{K}^{-1}[\mathbf{p}_c^i, 1]^T d_c^i(\mathbf{w})$ . Let  $\mathbf{T}_{c \rightarrow k1}^{\text{SLAM}}$  represent the relative camera pose of frame  $k1$  w.r.t. frame  $c$  obtained from pRGBD-SLAM. Using  $\mathbf{T}_{c \rightarrow k1}^{\text{SLAM}}$ , we can transfer the 3D point  $\mathbf{X}_c^i(\mathbf{w})$  from frame  $c$  to  $k1$  as:  $\mathbf{X}_{c \rightarrow k1}^i(\mathbf{w}) = \mathbf{T}_{c \rightarrow k1}^{\text{SLAM}} \mathbf{X}_c^i(\mathbf{w}) = [x_{c \rightarrow k1}^i(\mathbf{w}), y_{c \rightarrow k1}^i(\mathbf{w}), d_{c \rightarrow k1}^i(\mathbf{w})]^T$ .

Here,  $d_{c \rightarrow k1}^i(\mathbf{w})$  is the transferred depth of the  $i^{\text{th}}$  keypoint from frame  $c$  to frame  $k1$ . Following the above procedure, we can obtain the transferred depth  $d_{k1 \rightarrow c}^i(\mathbf{w})$  of the same  $i^{\text{th}}$  keypoint from frame  $k1$  to frame  $c$ . The symmetric depth transfer loss of the keypoint  $\mathbf{p}_c^i$  between frame pair  $c$  and  $k1$ , is the sum of absolute errors ( $\ell_1$  distance) between the transferred network-predicted depth  $d_{c \rightarrow k1}^i(\mathbf{w})$  and the existing network-predicted depth  $d_{k1}^i(\mathbf{w})$  in the target keyframe  $k1$ , and vice-versa. Mathematically, it can be written as:

$$\mathcal{T}_{c \leftrightarrow k1}^i(\mathbf{w}) = |d_{c \rightarrow k1}^i(\mathbf{w}) - d_{k1}^i(\mathbf{w})| + |d_{k1 \rightarrow c}^i(\mathbf{w}) - d_c^i(\mathbf{w})|. \quad (2.1)$$

Similarly, we can compute the symmetric depth transfer loss of the same  $i^{\text{th}}$  keypoint between frame pair  $c$  and  $k2$ , *i.e.*,  $\mathcal{T}_{c \leftrightarrow k2}^i(\mathbf{w})$ , and between  $k1$  and  $k2$ , *i.e.*,  $\mathcal{T}_{k1 \leftrightarrow k2}^i(\mathbf{w})$ . We accumulate the total symmetric transfer loss between frame  $c$  and  $k1$  in  $\mathcal{T}_{c \leftrightarrow k1}$ , which is the loss of all the common tracked keypoints and the points within the patch of size  $5 \times 5$  centered at the common tracked keypoints. Similarly, we compute the total symmetric depth transfer loss  $\mathcal{T}_{c \leftrightarrow k2}$  and  $\mathcal{T}_{k1 \leftrightarrow k2}$  between frame pair  $(c, k2)$ , and  $(k1, k2)$  respectively.

### 2.3.3.2 Depth Consistency Loss

The role of the depth consistency loss is to make depth network's prediction consistent with the refined depth values obtained from the pRGBD-SLAM. Note that depth values from pRGBD-SLAM undergo multiple optimization over wide baselines, hence are more accurate and capture long-term consistencies. We inject these long-term consistent depths from pRGBD-SLAM to depth network through the depth consistency loss. The loss for the frame  $c$  can be written as follows:

$$\mathcal{D}_c = \frac{\sum_{i \in \mathcal{X}} |d_c^i(\mathbf{w}) - d_c^i(\text{SLAM})|}{|\mathcal{X}|}. \quad (2.2)$$

### 2.3.3.3 Photometric Reconstruction Loss

Denote the relative camera pose of frame  $\mathcal{I}_{c-1}$  and  $\mathcal{I}_{c+1}$  w.r.t. current keyframe  $\mathcal{I}_c$  obtained from pRGBD-SLAM by  $\mathbf{T}_{c-1 \rightarrow c}^{\text{SLAM}}$  and  $\mathbf{T}_{c+1 \rightarrow c}^{\text{SLAM}}$  respectively. Using frame  $\mathcal{I}_{c+1}$ ,  $\mathbf{T}_{c+1 \rightarrow c}^{\text{SLAM}}$ , network-predicted depth map  $d_c(\mathbf{w})$  of the keyframe  $\mathcal{I}_c$ , and the camera intrinsic  $\mathbf{K}$ , we can synthesize the current frame  $\mathcal{I}_c$  [56, 55]. Let the synthesized frame be represented in the functional form as:  $\mathcal{I}_{c+1 \rightarrow c}(d_c(\mathbf{w}), \mathbf{T}_{c+1 \rightarrow c}^{\text{SLAM}}, \mathbf{K})$ . Similarly we can synthesize  $\mathcal{I}_{c-1 \rightarrow c}(d_c(\mathbf{w}), \mathbf{T}_{c-1 \rightarrow c}^{\text{SLAM}}, \mathbf{K})$  using frame  $\mathcal{I}_{c-1}$ . The photometric reconstruction error between the synthesized and the original current frame [51, 55, 186] is then computed as:

$$\mathcal{P}_c = pe(\mathcal{I}_{c+1 \rightarrow c}(d_c(\mathbf{w}), \mathbf{T}_{c+1 \rightarrow c}^{\text{SLAM}}, \mathbf{K}), \mathcal{I}_c) + pe(\mathcal{I}_{c-1 \rightarrow c}(d_c(\mathbf{w}), \mathbf{T}_{c-1 \rightarrow c}^{\text{SLAM}}, \mathbf{K}), \mathcal{I}_c), \quad (2.3)$$

where we follow [55, 56] to construct the photometric reconstruction error function  $pe(\cdot, \cdot)$ . Additionally, we adopt the more robust per-pixel minimum error, multi-scale strategy, auto-masking, and depth smoothness loss  $\mathcal{S}_c$  from [56].

Our final loss for fine-tuning the depth network at the depth refinement step is the weighted sum of narrow baseline losses (*i.e.*, photometric ( $\mathcal{P}_c$ ) and smoothness loss ( $\mathcal{S}_c$ )), and wide baseline losses (*i.e.*, symmetric depth transfer ( $\mathcal{T}_{c \leftrightarrow k1}$ ,  $\mathcal{T}_{c \leftrightarrow k2}$ ,  $\mathcal{T}_{k1 \leftrightarrow k2}$ ) and depth

consistency loss ( $\mathcal{D}_c$ ):

$$\mathcal{L} = \alpha \mathcal{P}_c + \beta \mathcal{S}_c + \gamma \mathcal{D}_c + \mu (\mathcal{T}_{c \leftrightarrow k1} + \mathcal{T}_{c \leftrightarrow k2} + \mathcal{T}_{k1 \leftrightarrow k2}). \quad (2.4)$$

## 2.4 Experimental Analysis

We conduct experiments to evaluate depth refinement and pose refinement steps of our self-improving framework with the state-of-the-arts in self-supervised depth estimation and RGB-SLAM based pose estimation respectively.

### 2.4.1 Datasets and Evaluation Metrics

**KITTI Dataset.** Our experiments are mostly performed on the KITTI dataset [52], which contains outdoor driving sequences. We further split KITTI experiments into two parts: one focused on depth refinement evaluation and the other on pose refinement. For depth refinement evaluation we train/fine-tune the depth network using the Eigen train split [36] which contains 28 training sequences and evaluate depth prediction on the Eigen test split [36] following the baselines [186, 172, 96, 176, 150, 189, 171, 116, 88, 20]. For pose refinement evaluation, we train/fine-tune the depth network using KITTI odometry sequences 00-08 and test on sequences 09-10 and 11-21. Note, for evaluation on sequences 09-10 we use the ground-truth trajectories provided by [52], while for evaluation on sequences 11-21, since the ground-truth is not available we use the pseudo ground-truth trajectories obtained by running stereo version of ORB-SLAM on these sequences.

**TUM RGB-D Dataset.** For completeness and to demonstrate the capability of our self-improving framework on indoor scenes, we evaluate on the TUM RGB-D dataset [131], which consists of indoor sequences captured by a hand-held camera. We use 6 of 8 *freiburg3* sequences to train/fine-tune the depth network and the remaining 2 for evaluation. We choose *freiburg3* sequences because only they have *undistorted* RGB images and ground truth to train/fine-tune and evaluate respectively.

**Metrics for Pose Evaluation.** For quantitative pose evaluation, we compute the Root Mean Square Error (*RMSE*), Relative Translation (*Rel Tr*) error, and Relative Rotation (*Rel Rot*) error of the predicted camera trajectory. Since monocular SLAM systems can only recover camera poses up to a global scale, we align the camera trajectory estimated by each method with the ground truth one using the EVO toolbox [58]. We then use the official evaluation code from the KITTI Odometry benchmark to compute the *Rel Tr* and *Rel Rot* errors for all sub-trajectories with length in  $\{100, \dots, 800\}$  meters.

**Metrics for Depth Evaluation.** For quantitative depth evaluation, we use the standard metrics, including the Absolute Relative (*Abs Rel*) error, Squared Relative (*Sq Rel*) error, *RMSE*, *RMSE log*,  $\delta < 1.25$  (namely *a1*),  $\delta < 1.25^2$  (namely *a2*), and  $\delta < 1.25^3$  (namely *a3*) as defined in [36]. Again, since the depths from monocular images can only be estimated up to scale, we align the predicted depth map with the ground truth one using their median depth values. Following [36] and other baselines, we also clip the depths to 80 meters.

**Note.** In all the tables, the best performance is shown in **bold** and the second best is underlined.

### 2.4.2 Implementation Details

We implement our framework based on Monodepth2 [56] and ORB-SLAM [102], *i.e.*, we use the depth network of Monodepth2 and the RGB-D version of ORB-SLAM for depth refinement and pose refinement respectively. We would like to emphasize, that our self-improving strategy is not specific to MonoDepth2 or ORB-SLAM. Any other depth network that allows to incorporate SLAM outputs and any SLAM system that can provide the desired SLAM outputs can be put into the self-improving framework. We set the weight of the smoothness loss term of the final loss (Eq. (2.4))  $\beta = 0.001$  similar as in [56] and  $\alpha, \gamma$ , and  $\mu$  to 1. The ablation study results on disabling different loss terms can be found in Tab. 2.2.

**KITTI Eigen Split/Odometry Experiments.** We pre-train MonoDepth2 using monocular videos of the KITTI Eigen split training set with the hyper-parameters as suggested in MonoDepth2 [56]. We use an input/output resolution of  $640 \times 192$  for pre-training/fine-tuning and scale it up to the original resolution while running pRGBD-SLAM. We use same hyperparameters as for KITTI Eigen split to pre-train/fine-tune the depth model on KITTI Odometry train sequences mentioned in Sec. 2.4.1. During a self-improving loop, we *discard* pose network of MonoDepth2 and instead use camera poses from pRGBD-SLAM.

Outlier Removal. Before running a depth refinement step, we run an outlier removal step on the SLAM outputs. Specifically, we filter out outlier 3D map points and the associated 2D keypoints that satisfy at least one of the following conditions: i) it is observed in less than 3 keyframes, ii) its reprojection error in the current keyframe  $\mathcal{I}_c$  is larger than 3 pixels.

Camera Intrinsic. Monodepth2 computes the average camera intrinsics for the KITTI dataset and uses it for the training. However, for our fine-tuning of the depth network, using the average camera intrinsics leads to inferior performance, because we use the camera poses from pRGBD-SLAM, which runs with different camera intrinsics. Therefore, we use different camera intrinsics for different sequences when fine-tuning the depth network.

For fine-tuning the depth network pre-trained on KITTI Eigen split training sequences, we run pRGBD-SLAM on all the training sequences, and extract camera poses, 2D keypoints and the associated depths from keyframes. For pRGBD-SLAM(RGB-D ORB-SLAM), we use the default setting of ORB-SLAM, except for the adjusted  $b$  described in Sec. 2.3.1. The same above procedure is followed for depth model pre-trained on KITTI Odometry training sequences. The average number of keyframes used in a self-improving loop is  $\sim 9K$  and  $\sim 10K$  for KITTI Eigen split and KITTI Odometry experiments respectively. At each depth refinement step, we fine-tune the depth network parameters with 1 epoch only, using learning rate  $1e-6$ , keeping all the other hyperparameters the same as pre-training. For both KITTI Eigen split and KITTI Odometry experiments we report results after 5 *self-improving loops*.

**TUM RGB-D Experiments.** For TUM RGB-D, we pre-train/fine-tune the depth network on 6 *freiburg3* sequences, and test on 2 *freiburg3* sequences. The average number of keyframes in a self-improving loop is  $\sim 3.5K$ . We use an input/output resolution of  $480 \times 320$  for pre-training/fine-tuning and scale it up to the original resolution while running pRGBD-SLAM. We report results after 3 *self-improving loops*.

Table 2.1: Depth evaluation result on KITTI Eigen split test set. M: self-supervised monocular supervision, and S: self-supervised stereo supervision, D: depth supervision. ‘-’ means the result is not available from the paper. pRGBD-Refined outperforms all the self-supervised monocular methods and several stereo only and combined monocular and stereo methods. Our results are after 5 self-improving loops.

Method	Train	Lower is better				Higher is better		
		Abs Rel	Sq Rel	RMSE	RMSE log	a1	a2	a3
Yang[172]	M	0.182	1.481	6.501	0.267	0.725	0.906	0.963
Mahjourian[96]	M	0.163	1.240	6.220	0.250	0.762	0.916	0.968
Klodt[71]	M	0.166	1.490	5.998	-	0.778	0.919	0.966
DDVO[150]	M	0.151	1.257	5.583	0.228	0.810	0.936	0.974
GeoNet[176]	M	0.149	1.060	5.567	0.226	0.796	0.935	0.975
DF-Net[189]	M	0.150	1.124	5.507	0.223	0.806	0.933	0.973
Ranjan[116]	M	0.148	1.149	5.464	0.226	0.815	0.935	0.973
EPC++[88]	M	0.141	1.029	5.350	0.216	0.816	0.941	0.976
Struct2depth(M)[20]	M	0.141	1.026	5.291	0.215	0.816	0.945	0.979
WBAF [185]	M	0.135	0.992	5.288	0.211	0.831	0.942	0.976
MonoDepth2-M (re-train) [56]	M	0.117	0.941	4.889	0.194	0.873	0.957	0.980
MonoDepth2-M (original) [56]	M	0.115	0.903	4.863	0.193	0.877	0.959	0.981
pRGBD-Refined	M	<b>0.113</b>	<b>0.793</b>	<b>4.655</b>	<b>0.188</b>	0.874	<b>0.960</b>	<b>0.983</b>
Garg[51]	S	0.152	1.226	5.849	0.246	0.784	0.921	0.967
3Net (R50)[112]	S	0.129	0.996	5.281	0.223	0.831	0.939	0.974
Monodepth2-S[56]	S	0.109	0.873	4.960	0.209	0.864	0.948	0.975
SuperDepth [111]	S	0.112	0.875	4.958	0.207	0.852	0.947	0.977
monoResMatch [145]	S	0.111	0.867	4.714	0.199	0.864	0.954	0.979
DepthHints [159]	S	0.106	0.780	4.695	0.193	0.875	<b>0.958</b>	<b>0.980</b>
DVSO[170]	S	<b>0.097</b>	<b>0.734</b>	<b>4.442</b>	<b>0.187</b>	<b>0.888</b>	<b>0.958</b>	<b>0.980</b>
UnDeepVO [79]	MS	0.183	1.730	6.570	0.268	-	-	-
EPC++ [88]	MS	0.128	0.935	5.011	0.209	0.831	0.945	<b>0.979</b>
Monodepth2-MS[56]	MS	<b>0.106</b>	<b>0.818</b>	<b>4.750</b>	<b>0.196</b>	<b>0.874</b>	<b>0.957</b>	<b>0.979</b>
Eigen[36]	D	0.203	1.548	6.307	0.282	0.702	0.890	0.890
Liu[84]	D	0.201	1.584	6.471	0.273	0.680	0.898	0.967
Kuznietsov[76]	DS	0.113	0.741	4.621	0.189	0.862	0.960	0.986
SVSM FT[88]	DS	0.094	0.626	4.252	0.177	0.891	0.965	0.984
Guo[60]	DS	0.096	0.641	4.095	0.168	0.892	0.967	0.986
DORN[49]	D	<b>0.072</b>	<b>0.307</b>	<b>2.727</b>	<b>0.120</b>	<b>0.932</b>	<b>0.984</b>	<b>0.994</b>

### 2.4.3 Monocular Depth Prediction / Refinement Evaluation

In the following, we evaluate the performance of our depth estimation on the KITTI Raw Eigen split test set, KITTI Odometry validation set and TUM RGB-D *freiburg3* sequences.

#### 2.4.3.1 Quantitative Depth Evaluation Results on KITTI Eigen’s Split Test Set

We show the depth evaluation results on the Eigen split test set in Tab. 2.1. From the table, it is evident that our refined depth model (pRGBD-Refined) outperforms all the competing monocular (M) unsupervised methods by non-trivial margins, including MonoDepth2-M re-trained depth model, and even surpasses the unsupervised methods with stereo (S) training, *i.e.*, Monodepth2-S, and combined monocular-stereo (MS) training,

*i.e.*, MonoDepth2-MS, in most metrics. Our method also outperforms several ground-truth depth supervised methods [36, 84]. The reason is probably that the aggregated cues from multiple views with wide baseline losses (*e.g.*, our symmetric depth transfer, depth consistency losses) lead to more well-posed depth recovery, and hence even higher accuracy than learning with the pre-calibrated stereo rig with smaller baselines. Further analysis is provided in Sec. 2.5. In Tab. 2.2 we show ablation experiment results, we run our framework for one self-improving loop and disable losses in the Eq. (2.4) one by one, the results in the table shows that, best performance is achieved only when using all the losses simultaneously for depth refinement.

Table 2.2: Ablation study on 1<sup>st</sup> self-improving loop. The best performance is in bold

Loss	Lower is better				Higher is better		
	Abs Rel	Sq Rel	RMSE	RMSE log	a1	a2	a3
w/o $\mathcal{D}_c$	<b>0.117</b>	0.958	4.956	0.194	0.862	0.955	0.980
w/o $\mathcal{T}_c$	0.118	0.955	4.867	0.194	0.872	0.957	0.980
w/o $\mathcal{P}_c$	<b>0.117</b>	0.942	4.855	0.194	<b>0.873</b>	<b>0.958</b>	0.980
all losses	<b>0.117</b>	<b>0.931</b>	<b>4.809</b>	<b>0.192</b>	<b>0.873</b>	<b>0.958</b>	<b>0.981</b>

### 2.4.3.2 Qualitative Depth Evaluation / Improvement Results

Fig. 2.6 shows some qualitative results in general, where pRGBD-Refined shows visible improvements at occlusion boundaries and thin objects. Fig. 2.7 shows some visual improvements in depth predictions of farther scene points. The reason for the improvements is the aggregated cues from multiple views with wider baselines (*e.g.*, our depth transfer and depth consistency losses) lead to more well-posed depth recovery.

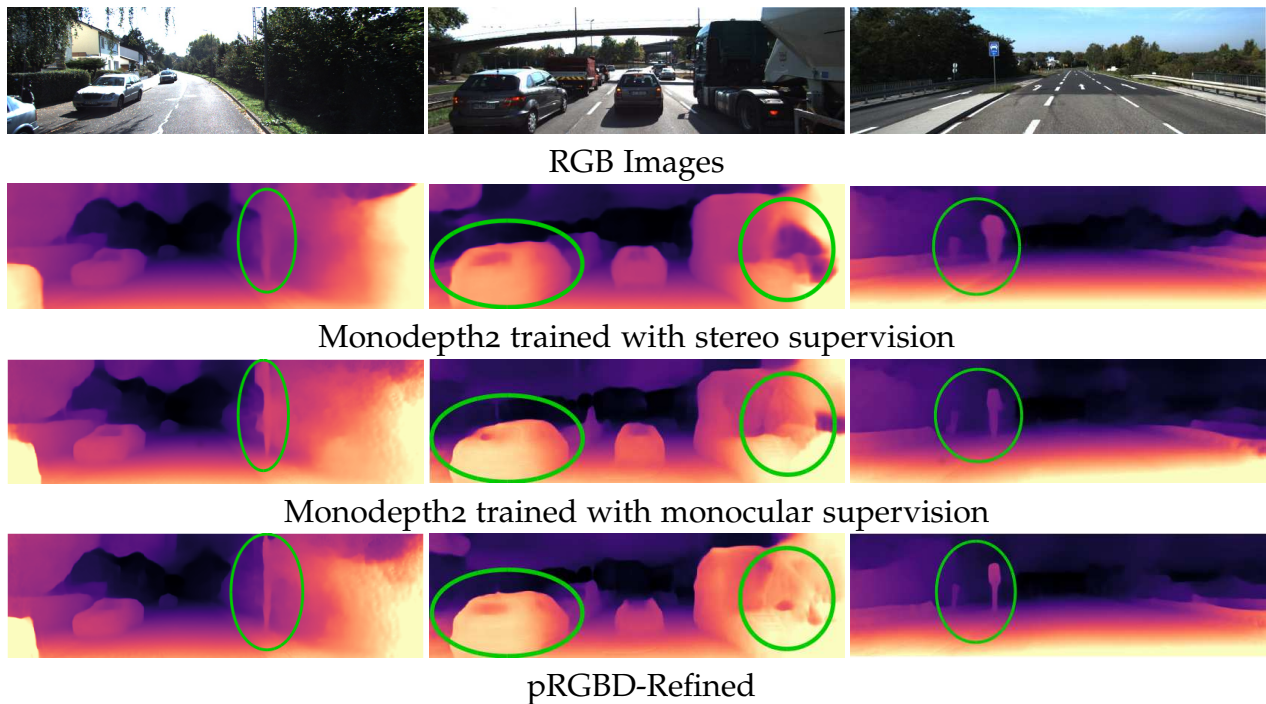


Figure 2.6: Qualitative depth evaluation results on KITTI Eigen's split test set.

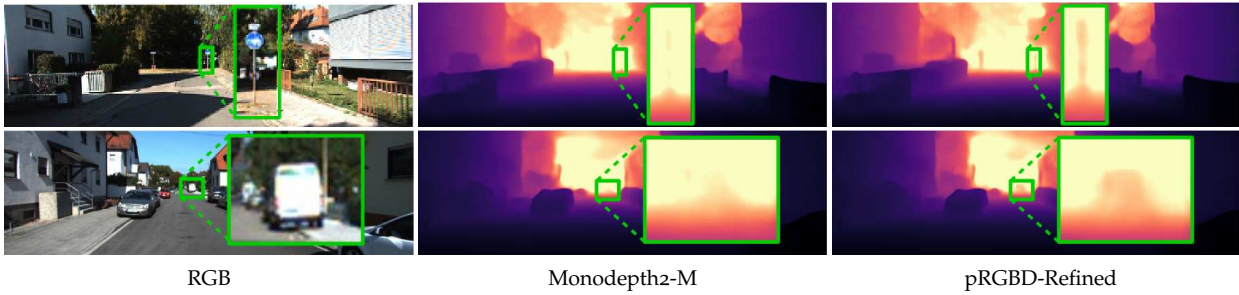


Figure 2.7: Qualitative depth improvement results in depth prediction of farther away scene points.

### 2.4.3.3 Quantitative Depth Evaluation Results on KITTI Odometry Validation Set (Sequences 09 and 10)

We evaluate the depth refinement step of our self-improving pipeline on KITTI Odometry sequences 09 and 10. We use sequence 00-08 for self-improving loops. The first block (*i.e.* MonoDepth2-M vs pRGBD-Refined) of the Tab. 2.3 shows the improved results after the depth refinement step. We also compare our method with a state-of-the-art depth refinement method DCNF [175]. **Note:** DCNF [175] uses *ground-truth* depths for pre-training the network, while our method uses only *unlabelled* monocular images, and still outperforms DCNF (see second block of the Tab. 2.3). The result shows that our self-improving framework with the wide-baseline losses (*i.e.*, symmetric depth transfer and depth consistency losses) improves the depth prediction.

Table 2.3: Qualitative depth evaluation on KITTI Odometry sequences 09 and 10. M: self-supervised monocular supervision for fine-tuning. ‘-’ means the result is not available from the paper. Our results are after 5 self-improving loops. Best results in each block is in bold.

Method	Train	Depth Cap	Lower is better				Higher is better		
			Abs Rel	Sq Rel	RMSE	RMSE log <sub>2</sub>	a1	a2	a3
MonoDepth2-M [56]	M	80	0.123	0.703	4.165	0.188	0.854	0.956	0.985
pRGBD-Refined	M	80	<b>0.121</b>	<b>0.649</b>	<b>3.995</b>	<b>0.184</b>	<b>0.853</b>	<b>0.960</b>	<b>0.986</b>
DCNF [175]	M	20	0.112	-	2.047	-	-	-	-
pRGBD-Refined	M	20	<b>0.098</b>	<b>0.242</b>	<b>1.610</b>	<b>0.145</b>	<b>0.906</b>	<b>0.978</b>	<b>0.993</b>

### 2.4.3.4 Quantitative Depth Evaluation Results on TUM RGB-D Sequences

The depth evaluation results on the two TUM *frieburg3* RGB-D sequences is shown in Tab. 2.4. Our refined depth model (pRGBD-Refined) outperforms pRGBD-Initial/Monodepth2-M in both sequences and all metrics.

Table 2.4: Quantitative depth evaluation results on two TUM *frieburg3* RGB-D sequences. pRGBD-Refined results are after 3 self-improving loops.

Method	TUM RGBD Sequences							
	Lower is better				Higher is better			a3
	Ab Rel	Sq Rel	RMSE	RMSElog	a1	a2		
pRGBD-Initial	0.397	0.848	1.090	0.719	0.483	0.722	0.862	
pRGBD-Refined	<b>0.307</b>	<b>0.341</b>	<b>0.743</b>	<b>0.655</b>	<b>0.522</b>	<b>0.766</b>	<b>0.873</b>	



### 2.4.3.5 Qualitative Depth Evaluation Results on TUM RGB-D Sequences

Some qualitative depth refinement results are presented in Fig. 2.8. It can be seen that the disparity between the depth values of nearby and farther scene points become clearer, *e.g.*, see depth around the two monitors.

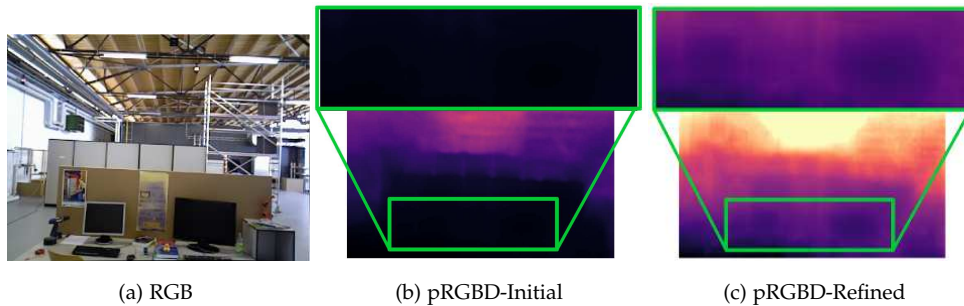


Figure 2.8: Qualitative depth evaluation results on TUM RGB-D sequences.

## 2.4.4 Monocular SLAM/Pose Refinement Evaluation

In this section, we evaluate pose estimation/refinement on the KITTI Odometry sequences 09 and 10, KITTI Odometry test set sequences 11-21, and two TUM *freiburg3* RGB-D sequences.

### 2.4.4.1 Quantitative Pose Evaluation Results on KITTI Odometry Sequences 09 and 10

We show the quantitative results on seqs 09 and 10 in Tab. 2.5. It can be seen that our pRGBD-Initial outperforms RGB ORB-SLAM [101] both in terms of RSME and Rel Tr. Our pRGBD-Refined further improves pRGBD-Initial in all metrics, which verifies the effectiveness of our self-improving mechanism in terms of pose estimation. The higher Rel Rot errors of our methods compared to RGB ORB-SLAM could be due to the high uncertainty of CNN-predicted depths for far-away points, which affects our rotation estimation [62]. In addition, our methods outperform all the competing supervised and self-supervised methods by a large margin, except for the supervised method of [168] with lower Rel Tr than ours on sequence 10. Note that we evaluate the camera poses produced by the pose network of Monodepth2-M [56] in Tab. 2.5, yielding much higher errors than ours.

### 2.4.4.2 Quantitative Pose Evaluation Results on KITTI Odometry Test Set

To facilitate the quantitative evaluation on this test set (*i.e.*, sequences 11-21), we use pseudo-ground-truth computed as mentioned in Sec. 2.4.1 to evaluate all the competing methods in Tab. 2.6. From the results, RGB ORB-SLAM fails on three challenging sequences due to tracking failures, whereas our pRGBD-Initial fails on two sequences and our pRGBD-Refined fails only on one sequence. Among the sequences where all

Table 2.5: Quantitative pose evaluation results on KITTI Odometry validation set. ‘-’ means the result is not available from the paper.

	Method	Seq. 09			Seq. 10		
		RMSE	Rel Tr	Rel Rot	RMSE	Rel Tr	Rel Rot
Supervised	DeepVO[157]	-	-	-	-	8.11	0.088
	ESP-VO[158]	-	-	-	-	9.77	0.102
	GFS-VO[169]	-	-	-	-	6.32	0.023
	GFS-VO-RNN[169]	-	-	-	-	7.44	0.032
	BeyondTracking[168]	-	-	-	-	<b>3.94</b>	<b>0.017</b>
	DeepV2D[134]	<b>79.06</b>	<b>8.71</b>	<b>0.037</b>	<b>48.49</b>	12.81	0.083
Self-Supervised	SfMLearner [186]	<b>24.31</b>	8.28	0.031	20.87	12.20	<b>0.030</b>
	GeoNet[176]	158.45	28.72	0.098	43.04	23.90	0.090
	Depth-VO[181]	-	11.93	0.039	-	12.45	0.035
	vidzdepth[96]	-	-	-	-	21.54	0.125
	UnDeepVO[79]	-	7.01	0.036	-	10.63	0.046
	Wang <i>et al.</i> [155]	-	9.88	0.034	-	12.24	0.052
	CC[116]	29.00	<b>6.92</b>	<b>0.018</b>	<b>13.77</b>	<b>7.97</b>	<b>0.031</b>
	DeepMatchVO[125]	<u>27.08</u>	9.91	0.038	24.44	12.18	0.059
	Li <i>et al.</i> [81]	-	8.10	0.028	-	12.90	0.032
	Monodepth2-M[56]	55.47	11.47	0.032	<u>20.46</u>	<b>7.73</b>	0.034
	SC-SfMLearner[12]	-	11.2	0.034	-	10.1	0.050
RGB ORB-SLAM	18.34	7.42	<b>0.004</b>	8.90	5.85	<b>0.004</b>	
pRGBD-Initial	12.21	4.26	0.011	8.30	<u>5.55</u>	0.017	
pRGBD-Refined	<b>11.97</b>	<b>4.20</b>	0.010	<b>6.35</b>	<b>4.40</b>	0.016	

the competing methods succeed, our pRGBD-Initial reduces the RMSEs of RGB ORB-SLAM by a considerable margin for all sequences except for sequence 19. After our self-improving mechanism, our pRGBD-Refined further boosts the performance, reaching the best results both in terms of RMSE and Rel Tr.

#### 2.4.4.3 Qualitative Pose Evaluation Results on KITTI Odometry

In this section, we show some qualitative pose evaluation results on KITTI Odometry dataset. Fig. 2.9(a) shows the camera trajectories estimated for sequence 09 by RGB ORB-SLAM, our pRGBD-Initial, and pRGBD-Refined. It is evident that, although all the methods perform loop closure successfully, our methods generate camera trajectories that align better with the ground truth. Fig. 2.9(b) shows qualitative comparisons on sequence 19. Note that RGB-SLAM fails after sometime. In Fig. 2.9 (c-e), all the three sequences our pRGBD-Refined aligned well with the ground-truth trajectory. Note that both RGB ORB-SLAM and our pRGBD-Initial fail on sequence 12, whereas our pRGBD-Refined succeeds, showing the enhanced robustness by our self-improving framework.

#### 2.4.4.4 Comparison with State-Of-The-Art SLAM Methods

In this section, we compare our pRGBD-Initial and pRGBD-Refined methods against state-of-the-art RGB SLAM methods, *i.e.*, Direct Sparse Odometry (DSO) [37], Direct Sparse Odometry with Loop Closure (LDSO) [50], and Direct Sparse Odometry in Dynamic Environments (DSOD) [89]. The results are shown in Tab. 2.7. From the results, it is evident that our pRGBD-Refined outperforms all the competing methods in Absolute Trajectory Error (RMSE) and Relative Translation (Rel Tr) Error. While the improvement in Absolute Trajectory Error (RMSE) and Relative Translation (Rel Tr) error is substantial, the performance in Relative Rotation (Rel Rot) is not comparable. The higher Rel Rot errors of our method compared to other RGB ORB-SLAM methods could be due to the

Table 2.6: Quantitative pose evaluation results on KITTI Odometry test set. Since the ground truth for the KITTI Odometry test set is not available we run Stereo ORB-SLAM[102] to get the complete camera trajectories and use them as the pseudo ground truth to evaluate. ‘X’ denotes tracking failure.

Seq	RGB ORB-SLAM			pRGBD-Initial			pRGBD-Refined		
	RMSE	Rel Tr	Rel Rot	RMSE	Rel Tr	Rel Rot	RMSE	Rel Tr	Rel Rot
11	14.83	7.69	<b>0.003</b>	<u>6.68</u>	<u>3.28</u>	0.016	<b>3.64</b>	<b>2.96</b>	<u>0.015</u>
13	<u>6.58</u>	<u>2.39</u>	<b>0.006</b>	6.83	2.52	0.008	<b>6.43</b>	<b>2.31</b>	<u>0.007</u>
14	<u>4.81</u>	<u>5.19</u>	<b>0.004</b>	<u>4.30</u>	<u>4.14</u>	<u>0.014</u>	<b>2.15</b>	<b>3.06</b>	<u>0.014</u>
15	3.67	1.78	<b>0.004</b>	<u>2.58</u>	<u>1.61</u>	0.005	<b>2.07</b>	<b>1.33</b>	<b>0.004</b>
16	6.21	2.66	<b>0.002</b>	<u>5.78</u>	<u>2.14</u>	0.006	<b>4.65</b>	<b>1.90</b>	<u>0.004</u>
18	6.63	2.38	<b>0.002</b>	<u>5.50</u>	<u>2.30</u>	0.008	<b>4.37</b>	<b>2.21</b>	<u>0.006</u>
19	<u>18.68</u>	4.91	<b>0.002</b>	23.96	<u>2.82</u>	0.007	<b>13.85</b>	<b>2.52</b>	<u>0.006</u>
20	9.19	6.74	<b>0.016</b>	<u>8.94</u>	<u>5.43</u>	0.027	<b>7.03</b>	<b>4.50</b>	<u>0.022</u>
12	X	X	X	X	X	X	<b>94.2</b>	<b>32.94</b>	<b>0.026</b>
17	X	X	X	<u>14.71</u>	<u>8.98</u>	<b>0.011</b>	<b>12.23</b>	<b>7.23</b>	<b>0.011</b>
21	X	X	X	X	X	X	X	X	X

high uncertainty of CNN-predicted depths for far-away points, which affects our rotation estimation [62]. However, if we compare Rel Rot error of pRGBD-Initial with the pRGBD-Refined, as depth prediction improves (see Tab. 2.3 MonoDepth2-M/pRGBD-Initial vs pRGBD-Refined) the Rel Rot error also improves (see Tab. 2.7).

Table 2.7: Comparison with state-of-the-art RGB SLAM methods on KITTI Odometry sequences 09 and 10. Here, ‘-’ means the result is not available from the original paper. \* denotes the result is obtained from [89].

Method	RMSE	Seq. 09		Rel Rot	RMSE	Seq. 10		Rel Rot
		Rel Tr	Rel Rot			Rel Tr	Rel Rot	
RGB ORB-SLAM[102]	18.34	7.42	<u>0.004</u>	8.90	5.85	<u>0.004</u>		
DSO[37]	74.29	72.27*	<b>0.002*</b>	16.32	80.81*	<b>0.002*</b>		
LDSO[50]	21.64	-	-	17.36	-	-		
DSOD[89]	-	13.85	<b>0.002</b>	-	13.53	<b>0.002</b>		
pRGBD-Initial	<u>12.21</u>	<u>4.26</u>	0.011	<u>8.30</u>	<u>5.55</u>	0.017		
pRGBD-Refined	<b>11.97</b>	<b>4.20</b>	0.010	<b>6.35</b>	<b>4.40</b>	0.016		

#### 2.4.4.5 KITTI Odometry Leaderboard Results

The KITTI Odometry leaderboard requires complete camera trajectories of all frames of all the sequences. Since we keep the default setting from ORB-SLAM in the section 2.4.4, causing tracking failures in a few sequences ( see Tab.2.4.4 ). The KITTI Odometry leaderboard requires the results of all sequences (i.e., sequences 11-21) for evaluation. Therefore, we change the default setting and increase the minimum number of inliers for adding keyframes from 100 to 500 so that our pRGBD-Refined succeeds on all sequences. We report the results of our pRGBD-Refined on the KITTI Odometry leaderboard in Tab. 2.8. Results show our method outperforms the competing monocular/LiDAR-based methods both in terms of relative translation and rotation errors.

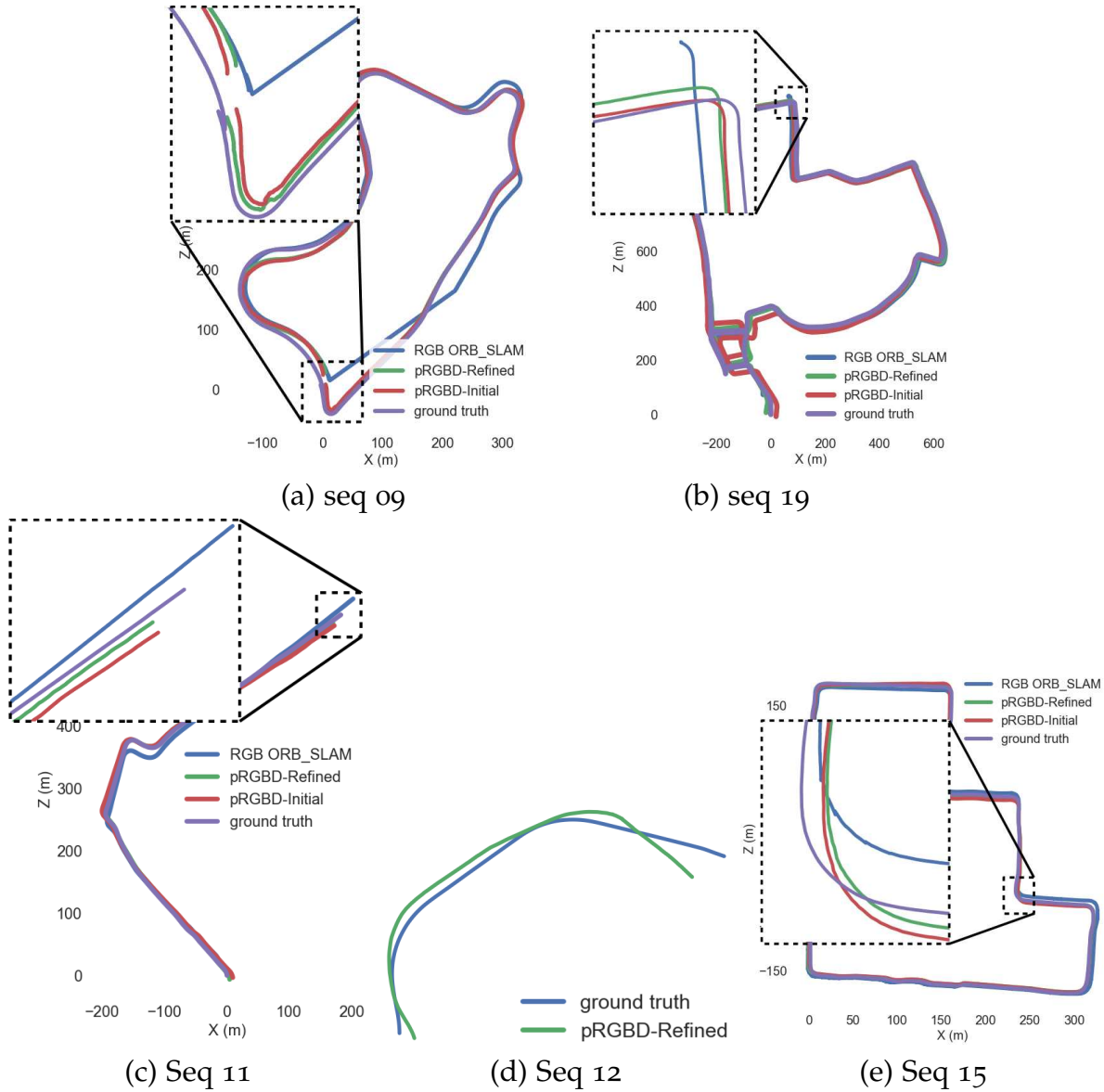


Figure 2.9: *Qualitative pose evaluation results on KITTI Odometry sequences. Note that both RGB ORB-SLAM fails in (b) and both RGB-SLAM and pRGBD-Initial fail in (d).*

#### 2.4.4.6 Quantitative Pose Evaluation Results on TUM RGB-D Sequences

Performance of pose refinement step on the two TUM RGB-D sequences is shown in Tab. 2.9. The result shows increased robustness and accuracy by pRGBD-Refined. In particular, RGB ORB-SLAM fails on walking\_xyz, while pRGBD-Refined succeeds and achieves the best performance on both sequences.

Table 2.9: *Quantitative pose evaluation results on two TUM frieburg3 RGB-D sequences. Note that RGB ORB-SLAM fail in walking\_xyz sequence.*

Seq	RGB ORB_SLAM	pRGBD-Initial	pRGBD-Refined
	RMSE	RMSE	RMSE
walking_xyz	X	<u>0.23</u>	<b>0.09</b>
large_cabinet_validation	1.72	<u>1.40</u>	<b>0.39</b>

Table 2.8: Quantitative pose evaluation results on KITTI Odometry leaderboard. Note that we use the estimated trajectories from ORB-SLAM2-S [102] for global scale alignment. The best performance is in bold.

Method	Rel Tr	Rel Rot
ORB-SLAM2-S [102]	1.70	0.0028
OABA [48]	20.95	0.0135
VISO2-M [53]	11.94	0.0234
BLO [149]	9.21	0.0163
VISO2-M+GP [53, 130]	7.46	0.0245
pRGBD-Refined	<b>6.24</b>	<b>0.0097</b>

#### 2.4.4.7 Qualitative Pose Evaluation Results on TUM RGB-D Sequences

Fig. 2.10(a) and Fig. 2.10(b) shows qualitative pose evaluation results on test sequences *walking\_xyz* and *large\_cabinet\_validation* respectively. The results, show the increased robustness and accuracy by pRGBD-Refined. In particular, RGB ORB-SLAM fails on *walking\_xyz*, while pRGBD-Refined succeeds and achieves the best performance on both sequences.

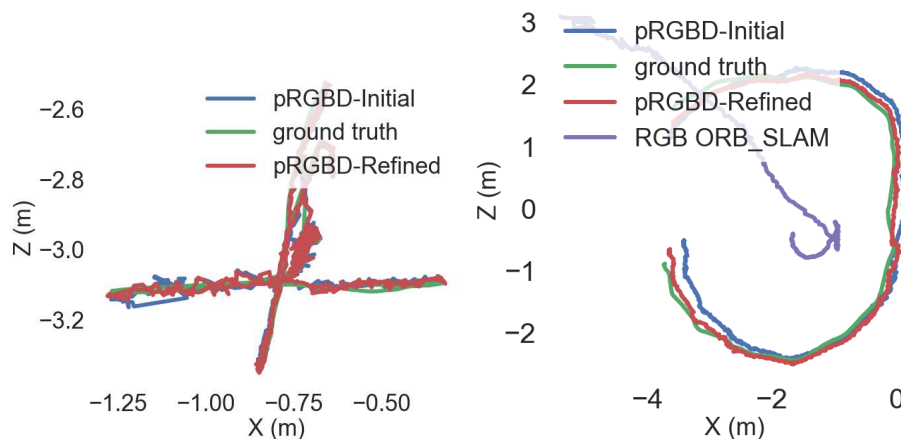


Figure 2.10: Qualitative pose evaluation results on TUM RGB-D sequences. Note that RGB ORB-SLAM fails in (a).

## 2.5 Analysis of Self-Improving Loops

In this section, we analyze the behaviour of three different evaluation metrics for depth estimation as defined in Sec. 2.4. The pose estimation is evaluated using the absolute trajectory pose error. In Fig. 2.11, we use the KITTI Eigen split dataset and report these metrics for each iteration of the self-improving loop. The evaluation metrics corresponding to the  $0^{th}$  self-improving loop are of the pre-trained MonoDepth2-M. We summarize the findings from the plots in Fig. 2.11 as below:

- A comparison of evaluation metrics of farther scene points (e.g. max depth 80) with nearby points (e.g. max depth 30) at the  $0^{th}$  self-improving loop shows that the

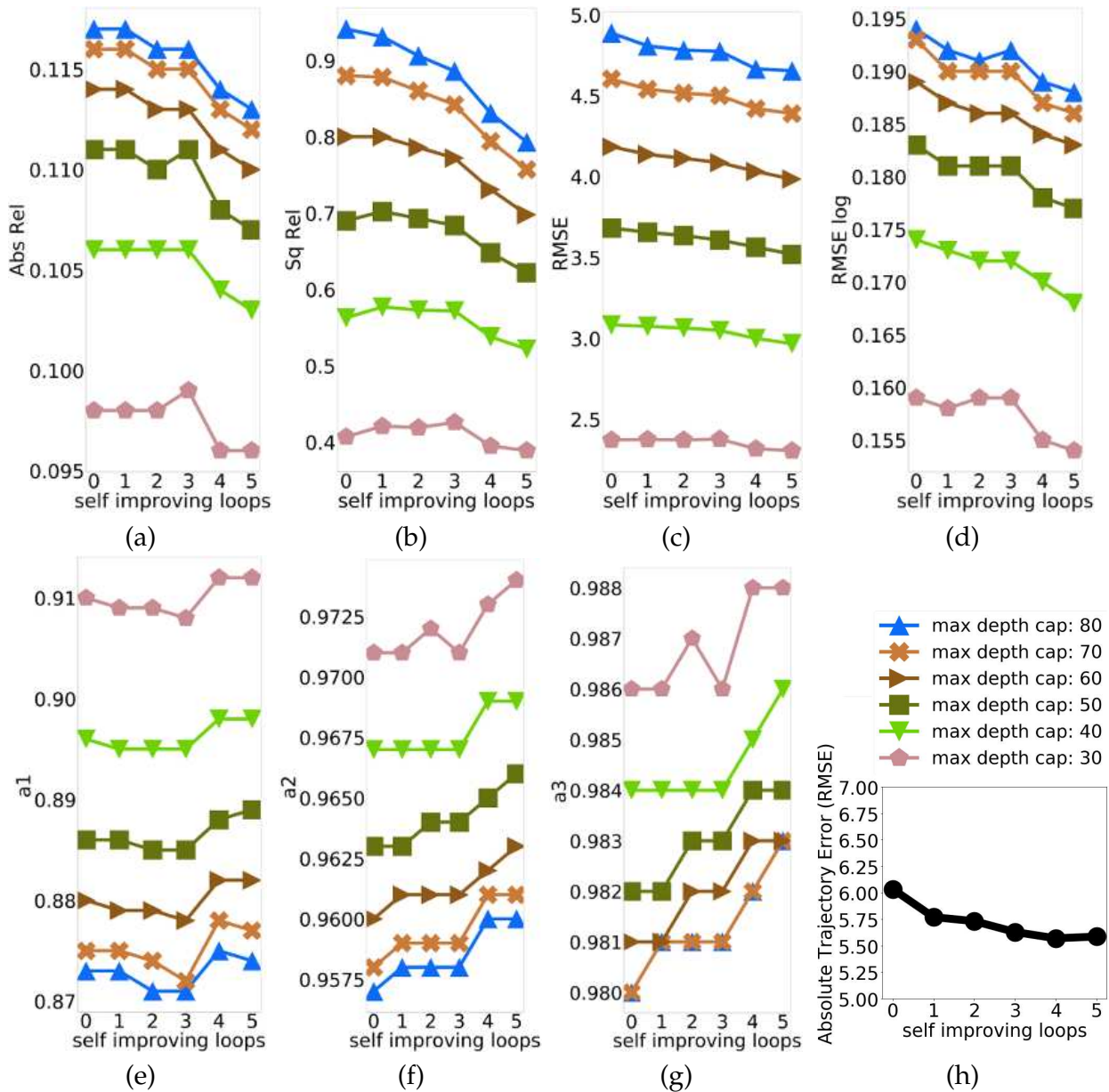


Figure 2.11: Depth/Pose evaluation metrics w.r.t. self-improving loops. (a). Absolute Relative (Abs Rel) (lower is better) (b). Squared Relative (Sq Rel) (lower is better) (c). RMSE (lower is better) (d). RMSE Log (lower is better), (e).  $a_1$  (higher is better), (f).  $a_2$  (higher is better), (g).  $a_3$  (higher is better) and (h) Absolute Trajectory Error (RMSE) (lower is better). Depth evaluation metrics in (a-g) are computed at different max depth caps ranging from 30-80 meters.

pre-trained MonoDepth2 performs poorly for farther scene points compared to nearby points.

- In the subsequent self-improving loops, we can see the rate of reduction in the Sq Rel and RMSE error is significant for farther away points compared to nearby points, *e.g.*, slope of error curves in Fig. 2.11(b-c) corresponding to max depth 80 is steeper than that of max depth 30. This validates our hypothesis of including wider baseline losses that help the depth network predict more accurate depth values for

farther points. Overall, our joint narrow and wide baseline based learning setup helps improve the depth prediction of both the nearby and farther away points, and outperforms MonoDepth2 [56].

- The error plot in Fig. 2.11(h) shows a decrease in pose error with self-improving loops and complements the improvement in depth evaluation metrics as shown in Fig.2.11(a)-(d). We terminate the self-improvement loop once there is no further improvement, *i.e.*, at the 5<sup>th</sup> iteration.

## 2.6 Discussion

We proposed a self-improving framework to couple geometrical and learning-based methods for 3D perception in this work. We demonstrated that the coupling of these two by leveraging the strengths of each mitigates the other’s shortcomings. Our proposed joint narrow and wide baseline-based self-improving framework, on the one hand, improved CNN-based depth prediction. On the other hand, our proposed feature-based Pseudo-RGBD SLAM framework has improved results compared to monocular RGB SLAM. The proposed framework is trained using only unlabeled monocular videos. The proposed geometric-CNN framework outperformed state-of-the-art self-supervised monocular and stereo depth prediction networks (e.g., Monodepth2), and feature-based monocular SLAM system (*i.e.*, ORB-SLAM).

A win-win situation has been achieved — both the monocular SLAM and depth prediction have been improved by a significant margin without any additional active depth sensor or ground-truth label. Currently, our self-improving framework only works in an off-line mode, so developing an on-line real-time self-improving system remains one of our future works. Another avenue for our future works is to move towards more challenging settings, e.g., uncalibrated cameras [187] or rolling shutter cameras [188]. The dataset (KITTI) we have trained and evaluated on have motions almost in 2D and also the range is also not very large. Hence, it would be interested to evaluate the proposed framework in a general 3D motion scenarios and understand its limitation and general challenges.





## 3 Robust Multiple Geometric Model Fitting

*“The elementary impressions of a visual world are those of surface and edge”*

– James Gibson, Perception of a Visual World (1950)

### 3.1 Introduction

Robust multiple model fitting plays a crucial role in many computer vision applications. The goal of a robust model fitting pipeline is to recover the underlying geometric structures present in the noisy data contaminated with outliers. In computer vision, frequently occurring geometric structures (or models) are planar homographies, fundamental matrices, vanishing points, lines, circles, ellipses and such. Geometric model fitting is often the backbone of many downstream applications including motion segmentation, 3D reconstruction, visual tracking, and image-based 3D modeling. An accurate estimation of these model parameters facilitates an interpretable and straightforward representation of scene geometry, rigid body motion or camera pose, and can aid scene understanding. Often, low-level techniques are employed to obtain feature point correspondences across different images, which are the typical observations in multi-model fitting. These low-level techniques are oblivious to the underlying scene geometry, and typically produce a significant number of outliers, i.e., data points that do not adhere to any genuine model instances.

Unlike single model fitting, the multi-model fitting problem has additional challenges, like the *unknown number of models*, different and unknown *inlier noise scale* for each structure (Fig. 3.1), and overlap in structures. Of these, the two most important ones are usually addressed by the user providing a ground-truth input or some other auxiliary information. A typical multi-model fitting pipeline first generates a set of model hypotheses, then identifies the inliers for each of the hypothesized models (i.e., the set of data points satisfying the hypothesized model), and finally selects the a subset of the hypothesized models that best explains the entire data.

We can divide multiple-model fitting approaches into two categories based on how the hypothesized models are used in the fitting process. The first is sequential fitting, the methods in this category operate in a iterative *hypothesize-and-verify* framework (e.g. MultiRANSAC[190], pbM-Estimator [22]), the second is simultaneous fitting, the methods in this category further categorized into *mode-seeking based* (e.g. Randomized Hough Transform[166], Mean Shift[28]), *clustering-based* (e.g. J-Linkage[144], T-Linkage[95]), *matrix factorization based* (e.e. RPA[93], NMU[137]), *optimization-based* (e.g. QP-MF[178]). The iterative hypothesize-and-verify approaches randomly generate a set of model hypotheses

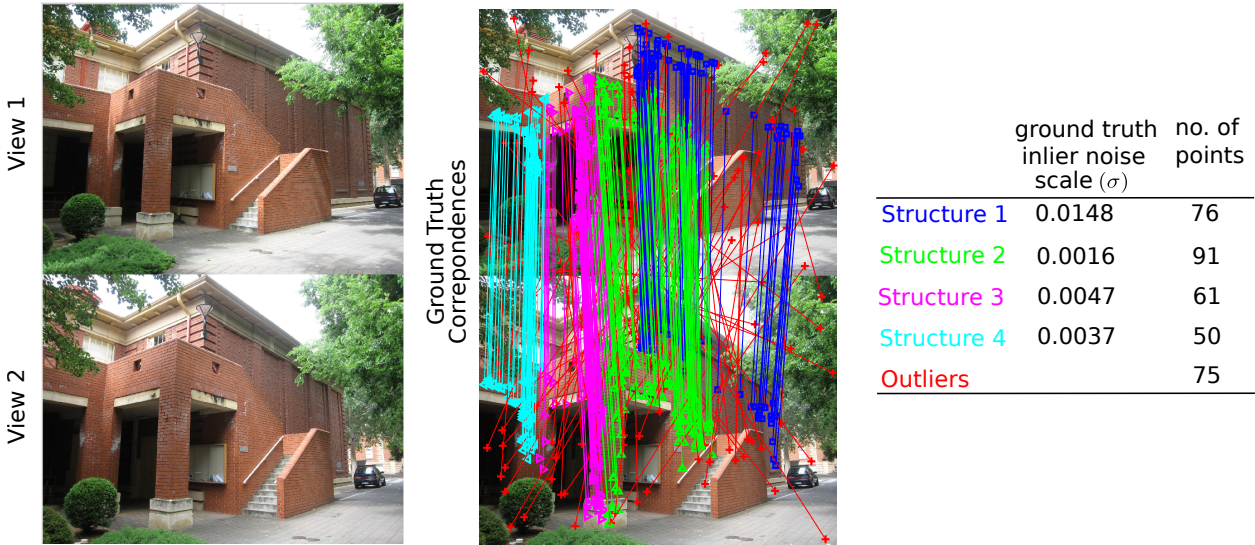


Figure 3.1: Johnsona, a multiple-homography fitting example from AdelaideRMF dataset [161]. Varying ground truth inlier noise scale and number of inliers.

fitted on randomly sampled minimal sample subsets of the data points. A hypothesis is selected from the set based on some robust criterion (e.g., number of inliers within a user-specified inlier threshold). The inliers of the selected hypothesis (model) are removed, and the whole process is repeated, starting from the random hypotheses generation. The second category methods take a set of generated hypotheses and apply different methods to simultaneously find multiple models. For example, the mode seeking based approaches apply a mode detection algorithm to the generated set model hypotheses in their parameter space. Each detected mode corresponds to a model. The clustering-based approaches leverage generated model hypotheses to construct a pairwise data similarity and use it to cluster the data points.

The iterative nature of the first category's methods makes the fitting process fast by iteratively reducing the original problem's size. However, sequential fitting has several crucial issues. The first issue is that the errors in the initial fits affect the fitting of other structures. The second issue is, in an overlapping structure scenario, removing inliers of the fitted structure reduces the inliers of the overlapped structures, which in turn reduces the chance of accurately fitting the overlapped structures in the subsequent iterations. Finally, it is non-trivial to find a stopping criterion that correlates with the actual number of models. On the other hand, the second category methods rely on the proportion of good hypotheses (from all genuine structures) present in the generated set of hypotheses. A significant proportion of bad hypotheses may overwhelm the model selection algorithm especially the mode and the cluster detection-based approaches.

The success of methods of both categories relies on obtaining a set of good hypotheses. Several guided sampling algorithms have been proposed to accelerate the generation of good hypotheses. Some use task specific input data properties like keypoint matching scores [47] in a two view planar/motion segmentation tasks to design a guided sampling strategy, while some use generic preference analysis approach [160, 24] to accelerate the hypothesis generation process. Out of these, preference analysis-based algorithms have shown remarkable performance. However, they have one crucial disadvantage of

operating in a time budget framework, i.e., run the hypothesis generation algorithm for a fixed time and collect the hypotheses generated so far. The user sets this time budget as a *reasonable guess*. There is no universal minimum time budget selection criterion that ensures a good proportion of hypotheses for all the underlying genuine structures. Generally, a guided or random sampling method intends to generate a large proportion of good hypotheses for the larger structures, reducing the chance of generating good hypotheses for the smaller structures within the allotted time budget. These methods ignored an important property of first category methods, i.e., adaptively reduce the original problem's size, i.e. once a model is fit, remove its inliers, and focus on the remaining structures. This adaptive nature helps to reduce the complexity and size of the original problem.

One of our main contributions is the adaptive guided hypothesis generation algorithm in the proposed multi-model fitting pipeline. We deviate from the time budget framework and overcome the shortcomings of the recent guided hypothesis generation methods for multi-model fitting. Our sampling algorithm uses a data-driven stopping criteria that ensures at least one good hypothesis is generated for each data point. We use kernel density estimate (KDE) to model the distribution of residual errors for a given hypothesis, which captures the local consensus between data points in the residual space. Accordingly, we refer to this KDE as the *Kernel Residual Density* (KRD) and apply it to the proposed hypothesis generation algorithm and other components of the proposed full multiple-model fitting pipeline.

We consider a multiple model fitting scenario, where the given data has multiple instances of a geometric model, corrupted by *noise* and *outliers*. The data points/observations that follow a particular model instance are inliers to that instance and act as a *pseudo-outliers* to other instances. The data points that do not follow any model instances are *gross-outliers*. In the most general setting of multi-model fitting, all of the following will be unknown: the number of model instances, the inlier noise scale which is typically different for each model instance, and the gross-outlier points that may constitute a significant fraction of all points. The goal of multiple model-fitting pipeline is to recover all genuine model instances and their corresponding inliers, while identifying all gross-outliers for subsequent rejection. Once the inliers are obtained, the model parameters can be estimated using standard regression techniques.

The single model fitting problem has been largely studied in the context of consensus maximization and often solved using heuristic methods like Random Sample Consensus [45] (RANSAC) and its variants or optimal global methods like [25]. However, the performance of these approaches critically depends on an accurate estimate of the inlier noise scale (or equivalently, the fraction of inliers), which is usually provided by the user. For single model fitting, this process can be automated by applying scale estimation methods [152, 151, 94] or by marginalizing over the scale variable [7]. However, in the case of the multi-model fitting, the problem becomes more challenging due to additional unknowns and model instances that may interfere with the estimation process. Many proposed methods like [110, 24, 160, 78, 77, 95, 93, 92] circumvent these challenges by assuming that the *number of models* or *inlier scale* or *both* are known a priori, thus introducing user dependency. In this work, we achieve automatic robust multi-model fitting by exploiting the kernel residual density, which is a key tool for differentiating between inliers and outliers. Our pipeline includes an automatic guided hypothesis

generation, inlier fraction estimation, model selection and point-to-model assignment procedures, and doesn't rely on the user to provide inlier thresholds and number of models. Inliers yield a small residual value and form a dense cluster around the regression surface, whereas outliers (or pseudo-outliers) can have arbitrarily large residuals but a much lower density. Geometrically, we can say that inliers are densely packed around the regression surface, while outliers are spread sparsely in the residual space (see Fig. 3.4). In this chapter, we present the Density Guided Sampling and Consensus (DGSAC) as an automatic pipeline for robust multi-model fitting. An illustration of the components of the DGSAC and its possible applications is shown in Fig. 3.2.

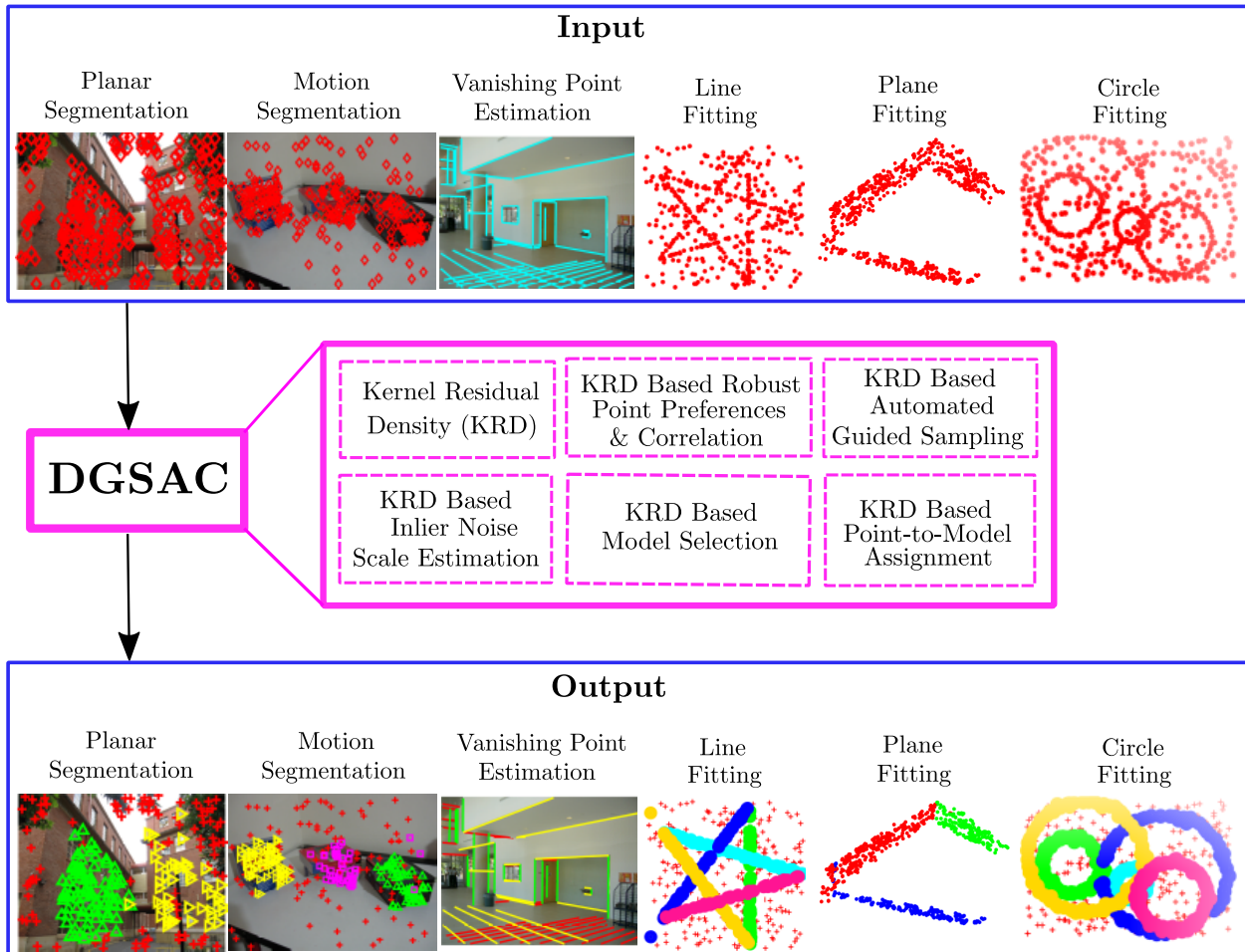


Figure 3.2: Density Guided Sampling and Consensus (DGSAC)

### 3.1.1 Contributions

Our specific contributions are summarized below:

- *Kernel Residual Density (KRD)*: We model the distribution of the residual errors using a kernel density estimate and refer to it as the Kernel Residual Density (KRD), which is central to various components of the multi-model fitting pipeline.
- *Kernel Density Guided Hypothesis Generation*: We present a novel iterative guided sampling approach driven by KRD based point correlations, which generates more

relevant model hypotheses from *all* inlier structures. To the best of our knowledge, we present the first non-time budget guided sampling approach, with a density-driven stopping criteria.

- *KRD based Inlier Noise Scale Estimation*: We present an inlier noise scale estimation approach based on simple yet effective information derived from residual dispersion and kernel residual density.
- *Greedy Method/Optimization Based Model Selection*: We propose two (*greedy* and *optimization based*) model selection algorithms. Both use the KRD to identify a unique model hypothesis from each genuine structure. The optimization based model selection is modeled as a quadratic program.
- *Multi-model fitting*: We combine the above modules to engineer an end-to-end automatic multi-model fitting solution that eliminates the need for user input (inlier threshold and number of models).
- *Extensive Evaluation*: We extensively evaluate DGSAC on a wide variety of tasks like motion segmentation, planar segmentation in the two-view images of a scene, vanishing point estimation/line segment classification in a single image, plane fitting to the 3D point cloud, and line and circle fitting to 2D points.

The remainder of the chapter is organized as follows. We first discuss the recent approaches for guided sampling and full multi-model fitting, and their limitations in Sec. 3.2.1. For completeness, we briefly discuss some of the traditional robust model fitting approaches and their limitations in Sec. 3.2.3. A formal problem statement is described in Sec. 3.3 followed by DGSAC Methodology in Sec. 3.4, the preliminaries are described in Sec. 3.5. We introduce Kernel Residual Density in Sec. 3.5.2, guided hypotheses generation in Sec. 3.5.5, inlier noise scale estimation in Sec. 3.5.9, model selection algorithms in Sec. 3.5.10 and point-to-model assignment in Sec. 3.5.11. The extensive experimental analysis is shown in Sec. 3.6.

## 3.2 Related Work

### 3.2.1 Guided Sampling Approaches

**Preference Analysis Based:** Most of the guided sampling strategies for multi-model fitting approaches exploit the concept of *preferences* [24, 161]. Multi-Guided Sampling (Multi-GS) [24] computes *hypothesis preferences* based on *ordered residuals* to derive a conditional distribution and use it for sampling minimal sample sets, i.e., the minimum number of samples required to fit a model (e.g., 2 samples are needed to model a line). Dynamic Hierarchical Filtering (DHF) [160] and Iterative Sample-and-Filter (ITKSF) [160] show improvement over Multi-GS [24]. Both DHF and ITKSF use *hypothesis and data preferences* to simultaneously sample and prune hypotheses. DHF [160] follows a per data point strategy, where its goal is to associate a hypothesis to each data point. These approaches operate in a time budget framework, and there is no clear stopping criteria or time budget constraint to ensure at-least one good hypothesis is generated for each genuine structure.

**Matching Score Based:** Recently, two approaches Efficient Guided Hypothesis Generation (EGHG) [78] and Unified Hypothesis Generation (UHG) [77] have been proposed. These approaches exploit matching scores of a feature matcher to derive the conditional distribution for sampling minimal sample sets. EGHG maintains two sampling loops, global and local. A set of good hypotheses maintained by a global sampling loop is used by local sampling to impose the epipolar constraints further, to improve the hypothesis generation. UHG first cluster the data points using T-Linkage [95] to prune out outliers. It uses the matching score of the clustered data points to derive conditional distribution for sampling minimal sample sets. However, the performance of T-Linkage [95], depends on the user-specified inlier threshold, which affects the performance of UHG. The approach of translating matching scores into inlier probability is not convincing because the local appearance based scores matching scores may result in false matches having a high matching score. Both EGHG [78] and UHG [77] require user-specified inlier threshold and operate in a time budget framework, which the user is expected to provide as a reasonable guess.

### 3.2.2 Full Multi-Model Fitting Approaches

Irrespective of the hypothesis generation process, in this section, we categorize recent best performing full end-to-end multi-model fitting approaches based on their model selection strategies and review them.

**Clustering Based:** J-Linkage [144] and T-Linkage [95] are two widely used clustering-based approaches in this category. J-Linkage [144] first represents the data points into their conceptual space and uses a linkage-based agglomerative clustering to cluster data points. This conceptual space is a hypothesis preference-based representation for data samples. T-linkage [95] is a variant of J-Linkage with a continuous conceptual representation of data points. T-Linkage [95] show improvement over J-Linkage [144]. However, the performance of both methods critically depends on the user-specified inlier threshold.

**Matrix Factorization Based:** Recently, matrix factorization based approaches like RPA [93] or NMU [137] have been proposed for model selection and have been shown to outperform clustering-based approaches. RPA assumes the knowledge of the inlier noise scale (albeit for the entire dataset) and the number of structures known a priori. It constructs a data point similarity matrix and decomposes it using symmetric NMF [74]. The decomposed matrix, along with the preference matrix, is used for final model selection. NMU outperforms RPA by enforcing an additional constraint of under-approximation. However, it also requires a user-specified noise scale estimate.

**Optimization Based:** This category of approaches form model selection as an optimization problem. QP-MF [178] formulate model selection as quadratic program. RCMSA [110] formulated the multi-model fitting problem in a simulated annealing and graph cut framework. It uses data preferences to construct a weighted graph. RansaCov [92] formulate model selection as a set-coverage problem. However, the performance of these approaches critically depends on the number of models and inlier threshold provided by a user.

### 3.2.3 Traditional Robust Model Fitting Methods

The early approaches to robust model fitting includes M-estimators [huber2004robust], least median of squares (LMS) [121], least trimmed squares (LTS) [121]. To understand their working and limitations, let us assume that we have an estimate of the model parameters, and a residual is defined as the difference between the observed and the fitted numerical value. The standard least-squares fitting method tries to find the model parameters by minimizing the squared residual of all the data points, which is unstable when outliers are present in the data.

An important metric to measure an estimator's robustness is the *breakdown point*; it is the fraction of outlying data points an estimator can handle before giving biased estimates. The least-squares method has a breakdown point of 0, as a single outlying data point can bias the estimates of the model parameters. To handle this, instead of minimizing the squared residuals of all the data points, the LMS minimizes the median of squared residuals. The LTS minimizes the first  $K$  smallest squared residuals. Generally, the LTS method is preferred over LMS due to its asymptotic efficiency (fast convergence rate) [121].

The M-estimator tries to reduce the effect of outliers. In contrast to directly minimizing the squared residual of all data points, it minimizes a function of a residuals. The function must be a symmetric, positive-definite, and have a unique minimum at zero. The formulation of M-estimator involves the derivative of the function with respect to the residual. This derivative is called the influence function [61]. The influence function measures the influence of a data point on the model parameters. An estimator is called to be robust when the influence of a single data point is insufficient to yield any significant bias in the model parameter estimates [119].

## 3.3 Problem Statement

**Notations.** A matrix is represented by capital and bold character *e.g.*  $\mathbf{H}$ . Its respective  $i^{\text{th}}$  row and  $j^{\text{th}}$  column are represented by its counterpart small bold as *e.g.*  $\mathbf{h}_i$  and  $\mathbf{h}^j$  respectively. The  $i^{\text{th}}$  row and  $j^{\text{th}}$  column element of the matrix  $\mathbf{H}$  is represented by small plain text as  $h_i^j$ . The sub-vector constituting the first  $k$  elements of the row vector  $\mathbf{h}_i$  is represented by  $\mathbf{h}_i^{1:k}$ . The sub-vector constituting the elements of the row vector  $\mathbf{h}_i$  whose indices are in the set  $w$  is represented by  $\mathbf{h}_i^{\{w\}}$ . The cardinality of a set  $w$  is denoted by  $|w|$ . The mean of the elements of a vector  $\mathbf{a}$  is denoted by  $mean(\mathbf{h}_i)$ .

**Multi-Model Fitting Problem.** We consider a multiple model fitting scenario. Given is the data set  $\mathbf{X} = \{x^j, j = 1, \dots, n\}$  of  $n$  data points originated from  $\kappa \geq 1$  instances of a geometric model. Here, we call  $x^j$  a *data point* in the abstract setting, it can be a point correspondence in homography fitting, or a 3D point in plane fitting, a line in vanishing point estimation, a 2D point in line fitting. Let the fraction of inliers of each model instance be denoted by  $f_i = \frac{|\mathcal{I}_i|}{n}$ ,  $i = 1, \dots, \kappa$ , where  $\mathcal{I}_i$  is an index set of inlier points of  $i^{\text{th}}$  model instance and  $f_0 = 1 - \sum_{i=1}^{\kappa} f_i$  denotes the fraction of gross outliers. Our goal is to output a set of  $\kappa$  tuples  $(\mathbf{h}_1, \mathcal{I}_1), (\mathbf{h}_2, \mathcal{I}_2), \dots, (\mathbf{h}_\kappa, \mathcal{I}_\kappa)$ , where  $\mathbf{h}_i$  and  $\mathcal{I}_i$  are the estimated model parameters and their corresponding inlier set respectively.

**Note:** We assume the number of models ( $\kappa$ ), inlier fractions ( $f_i$ 's) (or equivalently the respective inlier noise scales ( $\sigma_i$ 's)) are *unknown*. However, if provided this information can easily be incorporated in the proposed pipeline.

### 3.4 DGSAC Methodology

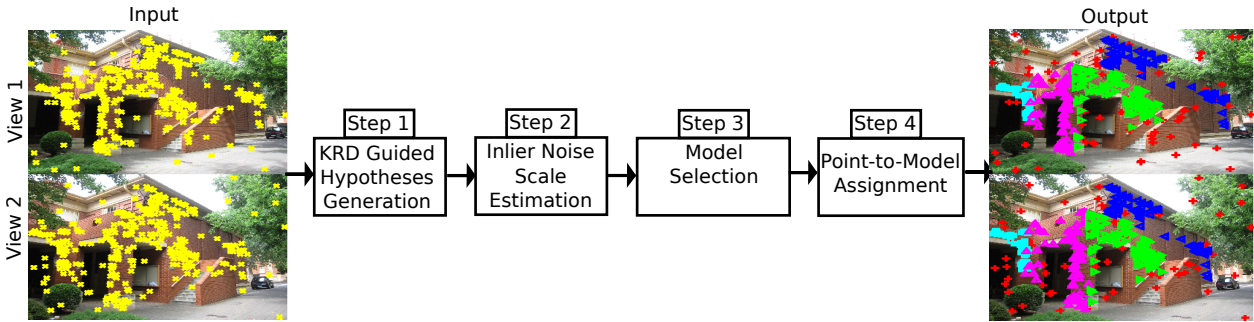


Figure 3.3: DGSAC pipeline consists of four major steps. Step 1: KRD guided hypothesis generation (Sec. 3.5.5). Step 2: Inlier noise scale estimation (Sec. 3.5.9). Step 3: Model selection (Sec. 3.5.10). Step 4: Point-to-Model assignment (Sec. 3.5.11). Here, we have shown an example of multiple homography estimation, the point membership in the output/ground truth is color coded. Irrespective of the fitting tasks (homography estimation, fundamental matrix estimation, vanishing point estimation, line, circle, and plane fitting), the four steps of the DGSAC pipeline remain the same.

DGSAC starts with generating a set of quality hypotheses using kernel residual density guided sampling (KDGS) (Sec. 3.5.5). The KDGS aims to generate good model hypotheses for each data point. The sampling process is guided by the conditional discrete inlier probability distribution derived from kernel residual density-based point correlation and potential good model hypotheses of each data point (Sec. 3.5.7). KDGS maintains a density-driven *explanation score* for each data point to ensure every point is explained by at least one good hypothesis. Once KDGS terminates, a model hypothesis is assigned to each data point based on the Kernel Residual Density (KRD) scores. We use the density and residual profiles of the hypotheses obtained from KDGS to estimate the inlier noise scale (Sec. 3.5.9), which we use to compute the goodness score of each model hypothesis. The generated hypotheses, along with their goodness score, are then fed into the model (hypothesis) selection algorithm (Sec. 3.5.10). The greedy algorithm (Sec. 3.5.10.3) starts with selecting a hypothesis with a high goodness score, maintaining a model diversity based on the overlap between estimated inlier sets of already selected and the remaining hypotheses (models). The optimization based model selection algorithm (Sec. 3.5.10.4) selects the final set of models (hypotheses) by solving a quadratic program. While the quadratic program's objective is to maximize the total goodness score, we further impose the penalty (Sec. 3.5.10.5) of selecting similar hypotheses (models). The final set of hypotheses obtained from model selection algorithm and their inlier set are then fed into a point-to-model assignment module (Sec. 3.5.11) to get the final output of the DGSAC pipeline *i.e.* a set of  $\kappa$  tuples as described in Sec. 3.3. The major steps of the DGSAC pipeline are shown in Fig. 3.3.

In the next sections, we describe each of the components of the DGSAC pipeline in detail.



## 3.5 Density Guided Sampling and Consensus

A model hypothesis  $\mathbf{h}_i$  can be obtained by fitting the model equation to a set of data points. Usually, this set contains the minimum number of data points required to estimate the model parameters, such a set is known as *minimal subset* (MSS). For example, if the geometric structure is a line, the cardinality of the minimal subset  $\eta$  is 2, and  $\mathbf{h}_i$  contains the value of slope and the line's intercept.

**Residual.** Residual of a data point  $x^j$  w.r.t. a hypothesis  $\mathbf{h}_i$  is a measure of disagreement of the data point w.r.t.  $\mathbf{h}_i$ . It is computed using a model specific residual function. Assume that we have generated a total of  $m$  model hypotheses  $\mathbf{H} = \{\mathbf{h}_i\}_{i=1}^m$ . We compute the residuals of all data points with respect to the hypothesis  $\mathbf{h}_i$  using the model specific residual function, which is defined as  $\psi(\mathbf{h}_i, x_j) : \mathbb{R}^d \rightarrow \mathbb{R}_+$  and store in a residual vector  $\mathbf{r}_i$  as given in equation (3.1).

$$\mathbf{r}_i = [r_i^1 = \psi(\mathbf{h}_i, x_1), \dots, r_i^n = \psi(\mathbf{h}_i, x_n)] \quad (3.1)$$

### 3.5.1 Hypothesis and Point preferences

Preference of a hypothesis is the rank ordering of all the data points based on some criteria *e.g.* residual [24] or residual density [139]. Analogously, the preference of a data point is the rank ordering of all model hypotheses. To compute *residual based hypothesis preferences* of a hypothesis  $\mathbf{h}_i$ , we find a permutation  $\mathbf{q}_i = [q_i^1, q_i^2, \dots, q_i^n]$  such that  $r_i^{q_i^1} \leq r_i^{q_i^2} \leq \dots \leq r_i^{q_i^n}$ . The ordered indices of data points, *i.e.*  $\mathbf{q}_i = [q_i^1, q_i^2, \dots, q_i^n]$  encodes the preferences of the hypothesis  $\mathbf{h}_i$ . Let  $\rho_i$  store the sorted residual vector as,  $\rho_i = [\rho_i^1, \rho_i^2, \dots, \rho_i^n] = [r_i^{q_i^1}, r_i^{q_i^2}, \dots, r_i^{q_i^n}]$ . Similarly we order  $\mathbf{r}^j$  in ascending order  $r_{l_1^j}^j \leq r_{l_2^j}^j \leq \dots \leq r_{l_m^j}^j$ . The ordered indices of data points, *i.e.*  $\mathbf{l}_j = [l_1^j, l_2^j, \dots, l_m^j]$  encodes the preferences of the data point  $x_j$ . Since, we use *residual* as a ordering criteria, therefore we refer  $\mathbf{q}_i$  and  $\mathbf{l}_j$  as *residual-based hypothesis* and *point* preferences respectively. In the later sections, we will discuss the kernel residual density (KRD) based preferences.

Preference analysis has been widely used in robust multi-model fitting. Previous approaches [24, 161, 160, 178] use residual as a criteria to derive point/hypothesis preferences. We demonstrate with an example in Fig. 3.4, a data point's top preference (a hypothesis to which it has the smallest residual) does not always correspond to a good model hypothesis (a model hypothesis that best describes the data point and the underlying model). The residual-based point's preference is conditioned only on the residual of the data point itself, and it ignores what other points in the neighborhood are preferring. We show point's preference derived from the proposed *Kernel Residual Density* (KRD) incorporates the local consensus information and provides more robust preferences. We demonstrate that density-based preferences truly capture the affinity of the data points towards the true underlying model.

In the next section, we describe the *kernel residual density* (KRD) for computing the KRD based points preferences. We first mathematically define the KRD and then discuss its importance in the context of model fitting. KRD is the most important component of the DGSAC pipeline, we will use it for computing point correlation (sec. 3.5.4), guiding

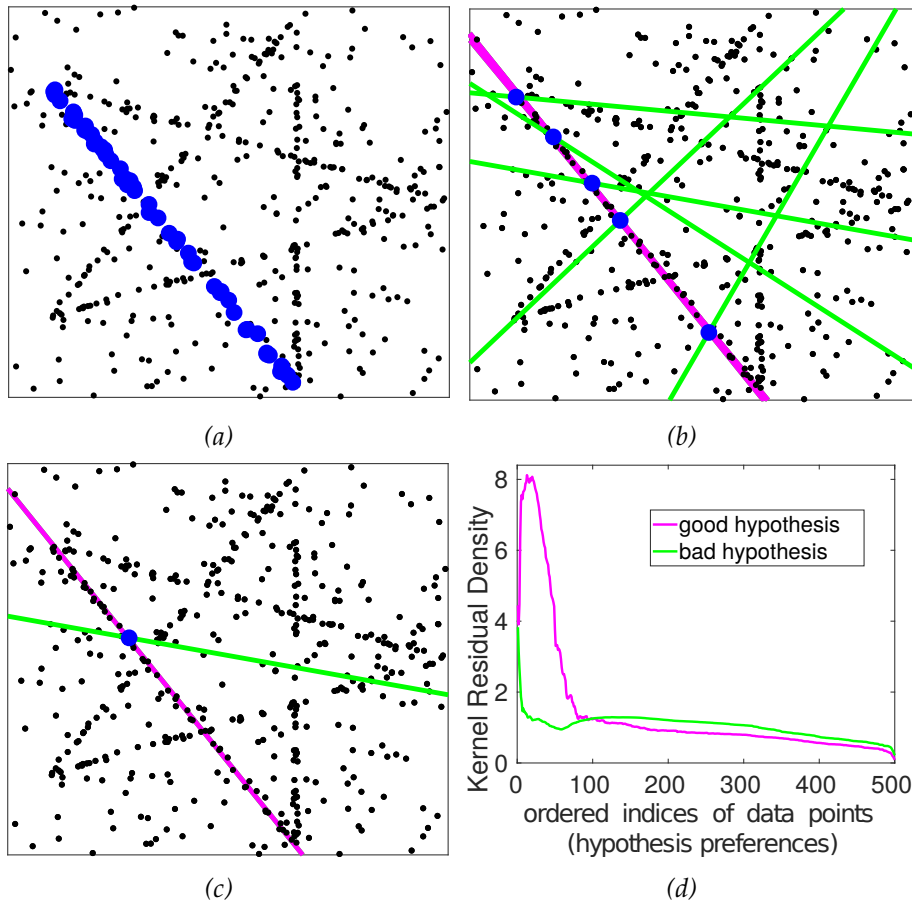


Figure 3.4: (a) Ground-Truth Structure, (b) Density based (*magenta*) and Residual based (*green*) top-1 preference of 5 inliers of the ground-truth structure (*blue*). It can be seen, the density based top preference of all 5 inliers is referring to the hypotheses generated from its true dense structure, while residual based top-1 preferences are the arbitrary hypotheses to which these data points have small residuals only. (c) a good (*magenta*) and a bad (*green*) hypothesis of the ground-truth structure shown in (a). (d) The Kernel Residual Density profile of the good and bad hypothesis shown in (c). It is expected for a good hypothesis to have high density around its regression surface (equivalently inliers should be densely packed around the regression surface). For a bad hypothesis density would be nearly flat except a very small peak around the regression surface due to small spurious structures.

hypothesis generation (sec. 3.5.5), inlier noise scale estimation (sec. 3.5.9) and model selection (sec. 3.5.10).

### 3.5.2 Kernel Residual Density (KRD)

The kernel residual density provides a nonparametric estimate of the distribution of the residuals and can be instrumental in differentiating inliers from outliers. Moreover, it can be used at various stages of a multiple-model fitting pipeline and is estimated directly from the data without any user input. We introduce the *kernel residual density*, which is mathematically defined as the variable bandwidth kernel density estimate at each data

point in the residual space of the hypothesis  $\mathbf{h}_i$  as follows:

$$d_i^j = \Phi(\mathbf{h}_i, x^j) = \frac{1}{n} \sum_{k=1}^n \frac{1}{b_i^j} K\left(\frac{r_i^j - r_i^k}{b_i^j}\right) \quad (3.2)$$

where,  $b_i^j$  is the variable bandwidth corresponding to the data point  $x^j$  in the one-dimensional residual space of  $\mathbf{h}_i$ . Here,  $K(\cdot)$  can be any kernel function that must be symmetric around the kernel origin. Kernel functions like the Gaussian kernel can be used, however, in this work, we only use Epanechnikov kernel [69] (3.3).

$$K(u) = \frac{3}{4}(1 - u^2) \quad (3.3)$$

s.t.  $|u| \leq 1$

The KRD captures the density of the data points in the 1-dimensional residual space. The critical factor in our KRD formulation is the choice of variable bandwidth. We choose the bandwidth  $b_i^j$  at the data point  $x^j$ , in the 1-dimensional residual space of hypothesis  $\mathbf{h}_i$  to be  $r_i^j$ , i.e., the corresponding residual. Our choice of variable bandwidth promotes a single dominant density peak around the regression surface of a model hypothesis. On the other hand, if we fix the bandwidth, it would likely give rise to many density peaks due to spurious local structures [153].

An example of kernel residual density computed for a good and bad hypothesis is shown in Fig. 3.4(d). The corresponding good and bad hypotheses are shown in Fig. 3.4(c). From the plots, we observe the following. 1) For a good hypothesis, the kernel residual density is large near the regression surface (within inlier region) and decreases as we move away from the regression surface (towards outlier region). 2) For a bad hypothesis, KRD is nearly flat throughout except a small and sharp peak near the regression surface. The sharp and small peak implies that a spurious hypothesis can only explain a very small number of points as opposed to a substantial and wide peak for a genuine model hypothesis.

### 3.5.2.1 Kernel Residual Density Scores

In the multiple model fitting scenario, the size (no. of inliers) and the inlier noise scale of the underlying genuine structures may vary significantly (see *johnsona* example in Fig.3.1 ). Consequently, the Kernel Residual Density profiles of good hypotheses of the respective structures would also vary significantly. The KRD profiles of the hypotheses fitted to ground truth inliers of the respective structures of *johnsona* example in Fig. 3.1 are shown in Fig. 3.5(a). We have also shown a hypothesis fitted on gross outliers (in red). The Kernel Residual Density of potential inliers of hypothesis corresponding to structure 2 is significantly larger than the KRD of probable inliers of other structures. The number of inliers of structure 2 is greater than that of other structures, and its inlier noise scale is also smaller than the others.

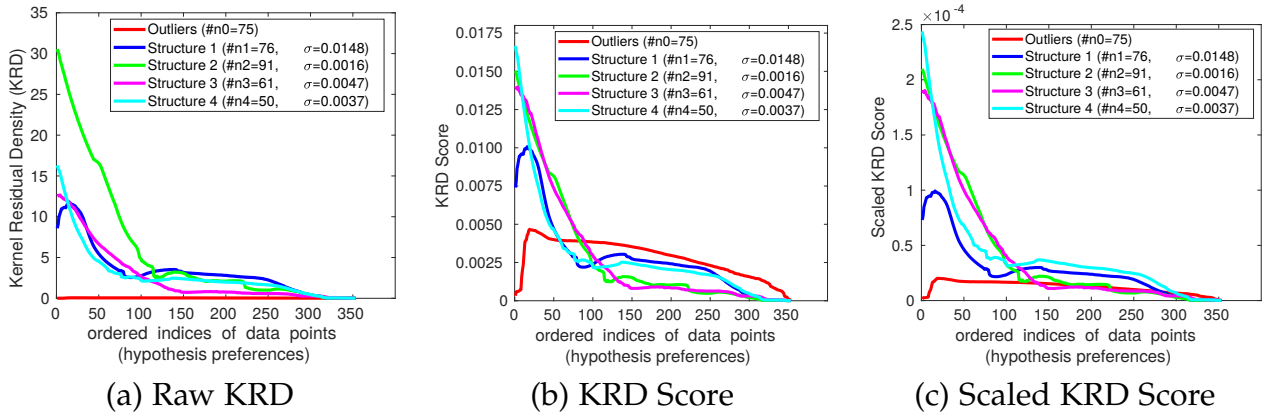


Figure 3.5: *Kernel Residual Density Scores of ground truth hypotheses fitted on inliers of the johnsona example shown in Fig. 3.1. (a) The raw KRD computed using Eq. 3.2 (b) normalized KRD scores computed using raw KRD in (a), and (c) scaled normalized KRD scores. Note: Here, we used ground truth to demonstrate the motivation behind scaled KRD scores. Throughout, in this work we assume the following are unknowns, (1).number of structures, (2). inliers noise scales, and (3). corresponding number of inliers.*

Motivated by this seemingly strong discriminative ability of KRD to distinguish between good and bad hypotheses, we extensively use KRD in all blocks of the proposed multi-model fitting pipeline. Moreover, to ensure the fitting process is not overwhelmed by the larger structures, we normalize and scale KRD to give equal importance to the potential inliers all generated hypotheses. We normalize the raw KRD to make it sum to 1 (Eq. 3.4). The normalized KRD profile is shown in Fig. 3.5(b). The normalization brings the KRD scores of all hypotheses in a similar range and makes them comparable across hypotheses. However, this also enhances the KRD score of bad hypotheses (in red). As we discussed in the previous section, a good hypothesis would always have a high disparity in the densities of the data points in the inlier and outlier regions. We use this property and scale the normalized KRD scores (Eq. 3.6). The scaling factor  $\pi$  for each hypothesis is the disparity in the mean of the normalized KRD scores of top- $\beta$  and bottom- $\beta$  preferences. The computation of scaling factor for  $i^{\text{th}}$  hypothesis (i.e.  $\pi_i$ ) is shown in Eq. 3.5, where  $w$ 's are density-based hypothesis preferences computed as follows,  $\mathbf{w}_i = [w_i^1, w_i^2, \dots, w_i^n]$  such that  $d_i^{w_i^1} \geq d_i^{w_i^2} \geq \dots \geq d_i^{w_i^n}$ . The scaled KRD scores are shown in Fig. 3.5(c). The scaling factor for a bad hypothesis (or outliers hypothesis) would always be smaller than a good hypothesis. Consequently, the difference between scaled KRD scores of data points in the inlier region of a bad and good hypothesis has increased after scaling.

$$d_i^j = d_i^j / \sum_{k=1}^n d_i^k, \quad \forall j \in [1, \dots, n] \quad (3.4)$$

$$\pi_i = \text{mean}(d_i^{w_i^1:w_i^\beta}) - \text{mean}(d_i^{w_i^{n-\beta}:w_i^n}) \quad (3.5)$$

$$d_i^j = d_i^j \times \pi_i, \quad \forall j \in [1, \dots, n] \quad (3.6)$$

Throughout this chapter we use scaled KRD scores. Henceforth, wherever we use the term "KRD" or "KRD scores", it will always refer to scaled normalized KRD scores.

Next, we use describe the process of computing point preferences using KRD scores.

### 3.5.2.2 KRD based point preferences.

Unlike residual based point preference in Sec. 3.5.1 where for each data point we ordered hypotheses in *increasing* order of their residual values, in density based point preferences we rank ordered hypotheses based on *decreasing* order of KRD scores. For each data point  $x^j$ , we find permutation  $\mathbf{v}^j = [v_1^j, v_2^j, \dots, v_m^j]$  such that  $d_{v_1^j}^j \geq d_{v_2^j}^j \geq \dots \geq d_{v_m^j}^j$ . The permutation vector  $\mathbf{v}^j$  encodes the KRD based point preference of  $j^{\text{th}}$  data point. *Note:* hypotheses are ordered in decreasing order of their density values.

In the next section, we discuss the benefits of Kernel Residual Density and its use in finding the point preferences.

## 3.5.3 Advantages of Kernel Residual Density

We discuss the benefits of Kernel Residual Density using a line fitting example in Fig. 3.4. We plotted the top-1 preferred hypothesis of 5 different inliers of a ground truth structure. The green hypotheses are the residual-based top-1 preference of  $j^{\text{th}}$  data point (*i.e.* hypothesis with index  $l_1^j$ ), while in magenta are the KRD based top-1 preferred hypothesis (*i.e.* hypothesis with index  $v_1^j$ ). We can see only density-based top-1 preferences are the good hypotheses that belong to a genuine geometric structure. This phenomenon is because the kernel density captures the *local consensus* information in the form of data points having similar residuals. Intuitively, density-based preference of a data point  $x^j$ , indirectly accounts all the data points lying within the variable bandwidth  $b_i^j$  of a hypothesis  $\mathbf{h}_i$  at data point  $x^j$ .

Another advantage of using kernel density is that it helps differentiate inliers and outliers. In the case of a good hypothesis, its inliers, due to their smaller residuals, form a dense cluster around the regression surface. In contrast, its outliers due to comparative larger residuals form a nearly flat region away from the regression surface. The above phenomenon can be verified from the kernel residual density profiles of a good and bad hypothesis shown in Fig. 3.4(d). We can see that the good hypothesis has a higher density around the regression surface and a nearly flat region as we move away from the regression surface. On the contrary, we can see a bad hypothesis has a nearly flat density profile throughout except for a small peak near the regression surface. In the following sections, we show the use of KRD in various components of DGSAC.

Next, we describe the density-based point correlation and discuss its advantage over residual-based point correlation.

## 3.5.4 KRD Based Point Correlation

Pairwise point correlation plays a key role in the sampling process [24, 161, 160, 178]. It is often used to derive the conditional sampling probabilities. We define the pairwise point

correlation as the fraction of overlapping top-T point preferences. The pairwise point correlation between data points  $x^i$  and  $x^j$  is computed by applying the intersection kernel over top-T preferences, as shown in Eq. 3.7, where,  $\mathbf{v}_{1:T}^i$  and  $\mathbf{v}_{1:T}^j$  are top-T preference of point  $x^i$  and  $x^j$  respectively (computed as described in Sec. 3.5.2.2).

$$c_i^j = \frac{\mathbf{v}_{1:T}^i \cap \mathbf{v}_{1:T}^j}{T} \quad (3.7)$$

The correlation values range between 0 and 1. A high correlation (*i.e.* 1) indicates the two data points have the same top-T preferences, hence it is most likely that they both belong to the same underlying structure. The degree of likeliness decreases as correlation decreases from 1 to 0. We capture KRD based pairwise point correlations of all pairs in the point correlation matrix (PCM)  $\mathbf{C}$ , where its elements are computed using Eq. 3.7.

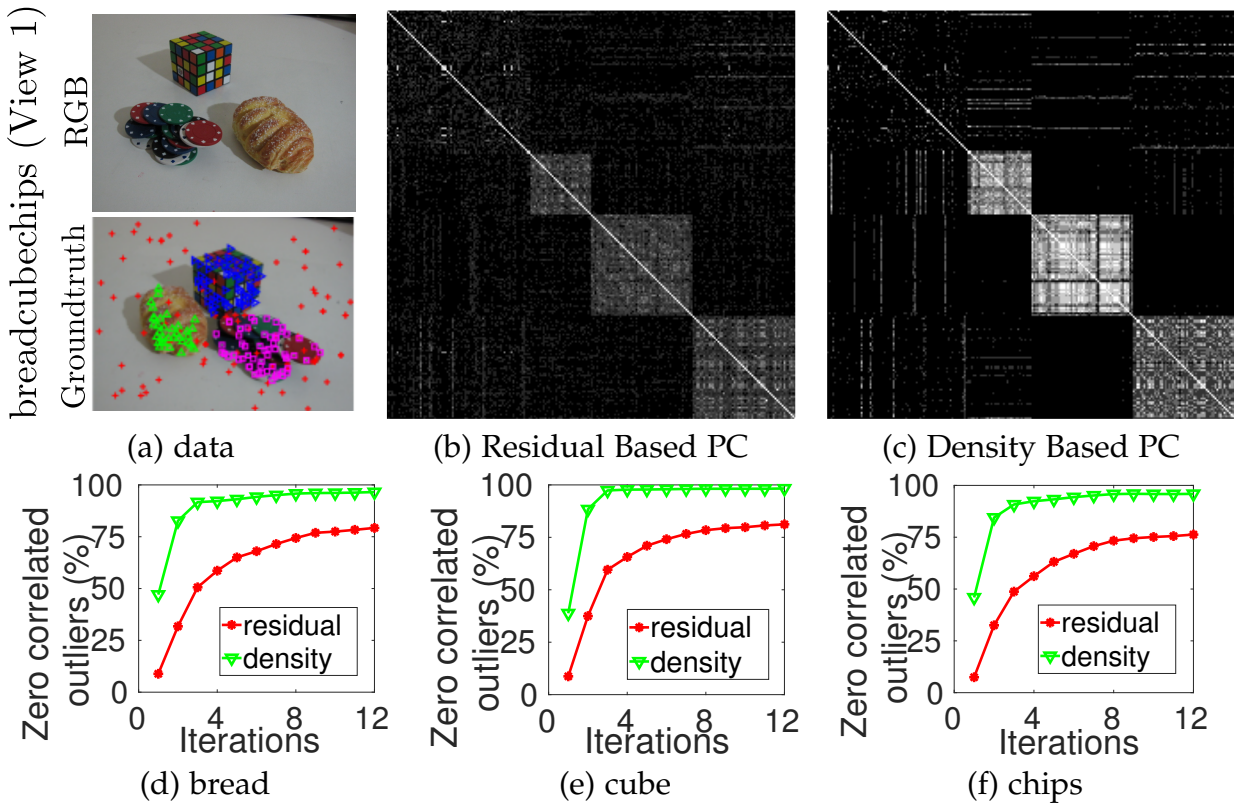


Figure 3.6: *Residual vs. KRD based point correlation: breadcubechips example of AdelaideRMF [161] dataset (b) Residual based and (c) Density based point correlation (PC) matrix. For better visualization the rows and columns in (b) and (c) are ordered by structure membership. For each of the three structures bread, cube and chips, plots in (d), (e) and (f) show percentage of uncorrelated outliers varying with iterations of (Algo. 1) respectively.*

### 3.5.4.1 Residual vs. KRD Based Point Correlation

Previous approaches [24, 161, 160, 178] have used residual-based point correlation in their multi-model fitting pipeline. It can be computed by applying the same intersection kernel over top-T preferences, but using residual-based preferences  $\mathbf{l}^i$  (see Sec. 3.5.1) instead of

density-based preferences  $\mathbf{v}^J$ . In this section, we compare residual and KRD based point correlation.

We fit multiple fundamental matrices to *breadcubechips* example of AdelaideRMF dataset [161], it has three (bread, cube and chips) genuine structures. A comparison between residual and KRD based PCM in Fig. 3.6. Ideally, a good point correlation computation strategy is the one, which provides a high correlation between a pair of inliers of the same structure and zero correlation otherwise. We show residual and KRD based point correlation matrix in Fig. 3.6(b) and 3.6(c) respectively, with brighter pixels indicating higher pairwise correlation. For better illustration, the points are ordered by structure membership, with the top set of rows being the gross outliers. For each three structures *bread*, *cube* and *chips*, the percentage of *uncorrelated* outliers (*i.e.* zeros pairwise correlation value computed using Eq. 3.7) varying with the iteration of KDGS Algo. 1 is plotted in 3.6(d), (e) and (f) respectively. At each iteration, the percentage is significantly larger for density-based PC than residual, which indicates that a high percentage of inliers are correlated with the other inliers of the same structure. Since the inliers of a hypothesis have a non-zero correlation with the other inliers and zero correlation with the outliers, any information derived from KRD based PC (*e.g.* conditional discrete inlier probability in Sec. 3.5.7) is robust to the outliers. The value of  $T$  is set after thorough empirical validation independently for both the residual and KRD based PCM. In the case of residual-based PCM, the choice of  $T = \lfloor 0.1m \rfloor$  is selected according to the prior work [24, 160], where  $m$  is the total number of hypotheses generated. In this work (DGSAC), we use *only* KRD based point correlation and the value of  $T$  is set to  $T = 5$  for *all* experiments in Sec. 3.6. Since the value of  $T$  is the function of  $m$  for residual based PC, its computation time will increase linearly with  $m$ , while for density-based PC it will be constant w.r.t.  $m$  because the value of  $T = 5$  is fixed. The constant computation time of pairwise density-based correlation makes it more advantageous over the residual-based PC.

### 3.5.5 KRD Guided Hypotheses Generation (KDGS)

Hypothesis generation is a critical component of a multi-model fitting pipeline. The quality of generated hypotheses significantly impacts the final result of the multi-model fitting pipeline, *i.e.*, classification of data point into their respective genuine structures. Generating a hypothesis is equivalent to sampling its corresponding minimal sample set (MSS) followed by a deterministic step of fitting the model equation to the MSS elements. Therefore, a guided sampling algorithm aims to bias the sampling process that generates a set of good hypotheses for each underlying genuine structure.

Most of the multi-structure guided sampling algorithms operate in the following framework: sample the first element from a uniform distribution, and the remaining elements from a conditional discrete inlier probability distribution, which is usually derived from the residual-based PCM [24, 160]. In the proposed Kernel Residual Density Guided Sampling (KDGS) algorithm, we also adopt a similar framework with the following modifications, (1). we sample the first element of MSS *deterministically* and the remaining from the conditional distribution. (2). the conditional inlier probabilities are derived using KRD based point correlation and the potentially good hypotheses (described later in Sec. 3.5.7) of the first sampled element. The complete KDGS algorithm is illustrated in (Algo. 1), which we describe below in detail.

### 3.5.6 Overview of the KDGS Algorithm

We adopt a per point good hypotheses generation strategy to ensure good hypotheses are generated for all big and small structures. KDGS maintains an index set  $\nu$  initialized by all data points, and generate hypotheses for each data point in  $\nu$ . Unlike previous guided sampling approaches, (1). we deterministically select the first element of each MSS, and (2). the remaining elements are sampled from conditional discrete inlier probability distribution derived from two different sources of information (described in Sec. 3.5.7). An *explanation score* is computed for each data point in the set  $\nu$ . Once the explanation score saturates, we remove the corresponding point from the set  $\nu$ . The saturation of the explanation score of a data point indicates that a sufficient number of good hypotheses have been generated for the particular data point (alternatively, we have generated sufficient hypotheses that can better explain the data point). Typically, the explanation scores of multiple data points saturates simultaneously, and we remove all of them collectively. As the algorithm progresses, the cardinality of the  $\nu$  set decrease and reach to zero. The algorithm stops when  $\nu$  is empty.

An extensive analysis of KDGS is provided in Sec.3.5.8.1, and a detailed comparison with state-of-the-art guided sampling approaches is provided in Sec. 3.6.1. We show, the change in  $|\nu|$ , *explanation scores* of inliers and *number of good hypotheses* of genuine structures as KDGS algorithm progresses.

In the next section, we first explain generating a single hypothesis for a data point by sampling a MSS starting from a single data point. Later in Sec. 3.5.8, we describe the complete flow of the KDGS algorithm.

### 3.5.7 Conditional Hypothesis Generation for a Data Point

#### 3.5.7.1 Potential good hypotheses of a data point

Assume that we already have generated a set of hypotheses in  $\mathbf{H}$ . Out of all the hypotheses in  $\mathbf{H}$ , which hypothesis will best describe the data point? A natural first response could be the hypothesis to which it has the smallest residual. However, as we have seen in Fig.3.4 and discussed in Sec. 3.5.8, the residual of a single data point alone does not always reflect its true affinity towards a hypothesis. Therefore, we argue that using residual alone as a measure is inadequate for finding a potential good hypothesis. Instead, we use measures that have some notion of local consensus around the data point. We find the first potential good hypothesis of  $j^{th}$  data point based on the KRD scores. We select a hypothesis to which the data point has a maximum KRD score. The second potential good hypothesis is selected based on the average of the top- $\beta$  smallest residuals, provided the point's residual itself lies within top- $\beta$ . We select the hypothesis which has minimum average residual. We select both the good potential hypotheses from a subset ( $\theta^j$ ) of the hypotheses. The subset contains those hypotheses to which the data point lies within their top- $\beta$  residual-based preferences. The set  $\theta^j$  is constructed as shown in Eq.3.8. Let the index for first and second potential good hypotheses be  $i_{den}$  and  $i_{res}$  respectively, and are computed as shown in Eq. 3.9 and 3.10 respectively.

$$\theta^j = \{i, \mid r_i^j \leq \rho_i^\beta\}, \quad \forall i \in \{1, \dots, m\} \quad (3.8)$$



$$i_{den} = \arg \max_{i \in \theta^j} d_i^j \quad (3.9)$$

$$i_{res} = \arg \min_{i \in \theta^j} \frac{\sum_{k=1}^{\beta} \rho_i^k}{\beta} \quad (3.10)$$

We fix  $\beta = 2\eta$ , it can be thought of the size of smallest *genuine* structure. For fitting tasks like line, plane, circle, and vanishing point, where  $\eta$  is very small ( $< 5$ ), we set  $\beta$  to be at-least 15.

### 3.5.7.2 Conditional inlier probability using potential good hypotheses

In the following, we describe the process of constructing the conditional inlier probability using potential good hypotheses of  $j^{th}$  data point.

We have seen in Sec. 3.5.2.1, the inlier have relatively higher KRD scores. Consequently, we can infer the probability of a data point being an inlier to a particular hypothesis is directly proportional to its KRD score w.r.t. the hypothesis. While the residual of a data point alone is not a very robust metric to find its affinity to a hypothesis (Sec. 3.5.3), the residual of an inlier would always be smaller than that of the outliers. Hence, we can infer, the probability of a data point being an inlier is inversely proportional to its residual. We use KRD scores of  $i_{den}$  hypothesis and compute the conditional discrete inlier probability

of each  $k^{th}$  data point as  $\frac{d_{i_{den}}^k}{\sum_{l \neq j} d_{i_{den}}^l}$ ,  $\forall k \in \{1, \dots, n\} \setminus \{j\}$ . Similarly, we use residual vector

of  $i_{res}$  hypothesis and compute conditional discrete inlier probability as  $\frac{\bar{r}_{i_{res}}^k}{\sum_{l \neq j} \bar{r}_{i_{res}}^l}$ ,  $\forall k \in \{1, \dots, n\} \setminus \{j\}$ , where  $\bar{r}_{i_{res}}^k = \frac{\max(r_{i_{res}})}{r_{i_{res}}^k}$ ,  $\forall k \in \{1, \dots, n\} \setminus \{j\}$ . For each  $k^{th}$  data point, we

compute the joint conditional discrete inlier probability as  $s_k^j = \frac{d_{i_{den}}^k}{\sum_{l \neq j} d_{i_{den}}^l} \times \frac{\bar{r}_{i_{res}}^k}{\sum_{l \neq j} \bar{r}_{i_{res}}^l}$ . We

stack inlier probability of all the data points in the vector  $\mathbf{s}^j = [s_1^j, s_2^j, \dots, s_n^j]$ , with  $s_j^j = 0$  and normalize it to sum  $\mathbf{1}$ , i.e.  $\sum_k s_k^j = 1$ . Let  $\mathbf{S}$  be the conditional inlier probability matrix, where each column corresponds to the conditional discrete inlier probability distribution of the respective data point i.e.  $\mathbf{S} = [\mathbf{s}^1, \dots, \mathbf{s}^j, \dots, \mathbf{s}^n]$ .

The vector  $\mathbf{s}^j$  is conditioned on  $j^{th}$  data point's potential good hypotheses. We use  $\mathbf{s}^j$  along with the pairwise point correlation of  $j^{th}$  data point with others (captured in vector  $\mathbf{c}^j$ ) to derive the final conditional discrete inlier probability distribution ( $\mathcal{P}$ ), and use it for sampling the elements of MSS for  $j^{th}$  data point. We intentionally set  $s_j^j = 0$  to enforce the sampling without replacement (discussed later in Sec. 3.5.7.4), as we deterministically select the  $j^{th}$  data point in the corresponding MSS.

### 3.5.7.3 Explanation score

We define an explanation score of  $j^{\text{th}}$  data point i.e.,  $\tau^j$  to be the average KRD score of  $j^{\text{th}}$  data point to the hypotheses in the set  $\theta^j$ . The  $\tau^j$  is computed as

$$\tau^j = \frac{\sum_{i \in \theta^j} d_i^j}{|\theta^j|} \quad (3.11)$$

Note that as the algorithm progresses, the set  $\theta^j$  changes dynamically and can be different for different data points. The saturation of  $\tau^j$  indicates that the KRD score of data point w.r.t. hypotheses in  $\theta^j$  has achieved approximately the maximum value. This indicates that the  $j^{\text{th}}$  data point now has sufficient hypotheses in the set  $\theta^j$  that can better explain it. Once the  $\tau^j$  achieves its saturation point, we remove it from the set  $\nu$ .

Next, we describe the conditional sampling process for generating a hypothesis for a single data point.

### 3.5.7.4 Conditional Sampling

Let  $\mathcal{M}$  be the minimal sample set of the hypothesis to be generated for  $j^{\text{th}}$  data point.

$$\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_\eta\} \subset \{1, 2, \dots, n\} \quad (3.12)$$

We deterministically select the  $j^{\text{th}}$  data point itself as the first element of the set  $\mathcal{M}$ , i.e.  $\mathcal{M}_1 = j$ . The remaining elements are sampled from conditional discrete inlier probability distribution. Let the second element of the set  $\mathcal{M}$  i.e.,  $\mathcal{M}_2$  is sampled from the conditional inlier probability distribution  $\mathcal{P}_1$ , i.e.  $\mathcal{M}_2 \sim \mathcal{P}_1$ . Similarly the  $(i+1)^{\text{th}}$  element is sampled from the conditional discrete inlier probability distribution  $\mathcal{P}_i$ . We construct the final conditional discrete inlier probability distribution  $\mathcal{P}_i$  using two sources of information, (1). KRD based point correlation ( $c^j$ ), and (2). conditional discrete inlier probability distribution ( $s^j$ ) derived from potential good hypotheses of  $j^{\text{th}}$  data point. The conditional inlier probability of  $k^{\text{th}}$  data point in the discrete probability distribution  $\mathcal{P}_i$  is denoted by  $\mathcal{P}_i(k)$ , and is computed as

$$\mathcal{P}_i(k) := \begin{cases} \mathcal{N}_i c_k^j \times s_k^j & \text{if } k \notin \{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_i\} \\ 0, & \text{otherwise} \end{cases} \quad (3.13)$$

$$\mathcal{N}_i = \frac{1}{\sum_{k \notin \{\mathcal{M}_1, \dots, \mathcal{M}_i\}} c_k^j \times s_k^j} \quad (3.14)$$

The  $\mathcal{N}_i$  in Eq. 3.13 is the normalization constant that ensures  $\mathcal{P}_i$  is a valid discrete probability distribution. The *if* and *otherwise* condition in Eq. 3.13 ensures sampling without replacement. The  $(i+1)^{\text{th}}$  element of the minimal sample set is sampled from  $i^{\text{th}}$  conditional inlier probability distribution as

$$\mathcal{M}_{i+1} \sim \mathcal{P}_i \quad (3.15)$$

The conditional sampling is achieved by using values in  $\mathcal{P}_i$  as sampling weights. For example, if  $\mathcal{P}_i(k) > \mathcal{P}_i(l)$ , then  $k^{\text{th}}$  data point is more likely to be sampled than  $l^{\text{th}}$  data point. The  $c_k^j$  in Eq. 3.13 is the pairwise KRD based point correlation of  $k^{\text{th}}$  data point with  $j^{\text{th}}$  data point. The  $s_k^j$  is the conditional inlier probability of  $k^{\text{th}}$  data point conditioned on  $j^{\text{th}}$  data point's potential good hypotheses.

---

**Algorithm 1: KRD Guided Sampling (KDGS)**


---

```

1 Input:  $\mathbf{X}, \eta$ , Output:  $\mathbf{H}, \mathbf{R}, \mathbf{D}$ 
2 Initialization:  $\mathbf{C} = \mathbb{1}_{n \times n}$ ,  $\mathbf{H} = \mathbf{R} = \mathbf{D} = \emptyset$ ,  $\tau_{prev}^k = \tau_{curr}^k = 0 \quad \forall k \in \{1, \dots, n\}$ 
3  $v = \{1, \dots, n\}$ ,  $\mathbf{S} = \{s_k^j\}$ ,  $\forall j, k \in \{1, \dots, n\}$ , where  $s_k^j = \begin{cases} \frac{1}{n-1}, & \text{if } j \neq k \\ 0, & j = k \end{cases}$ 
4 while  $v \neq \emptyset$  do
5    $[\widehat{\mathbf{H}}, \widehat{\mathbf{R}}, \widehat{\mathbf{D}}] = \text{generateHyps}(\mathbf{C}, v, \eta, \mathbf{S})$  ▷ Algorithm 2.
6    $\mathbf{H} = \{\mathbf{H} \cup \widehat{\mathbf{H}}\}$ ,  $\mathbf{R} = \{\mathbf{R} \cup \widehat{\mathbf{R}}\}$ ,  $\mathbf{D} = \{\mathbf{D} \cup \widehat{\mathbf{D}}\}$ 
7   for  $i = 1$  to  $m$  (no. of hypotheses in  $\mathbf{H}$ ) do
8      $d_i^j = d_i^j / \sum_{k=1}^n d_i^k$ ,  $\forall j \in \{1, \dots, n\}$ 
9      $\pi_i = \text{mean}(d_i^{w_i^1: w_i^\beta}) - \text{mean}(d_i^{w_i^{n-\beta}: w_i^n})$ 
10     $d_i^j = d_i^j \times \pi_i$ ,  $\forall j \in \{1, \dots, n\}$  ▷ scaled KRD scores. Sec.3.5.2.1.
11  end
12   $\mathbf{V} = \text{KRDPointPreferences}(\mathbf{D})$  ▷ point preferences. Sec. 3.5.2.2.
13  for  $j = 1$  to  $n$  do
14     $\theta^j = \{i, \text{ s.t. } r_i^j \leq \rho_i^\beta\}$ ,  $\forall i \in \{1, \dots, m\}$  ▷ Sec. 3.5.7.2
15     $i_{den} = \arg \max_{i \in \theta^j} d_i^j$  ▷ first potential good hypothesis.
16     $i_{res} = \arg \min_{i \in \theta^j} \frac{\sum_{k=1}^\beta \rho_i^k}{\beta}$  ▷ second potential good hypothesis.
17     $\bar{r}_{i_{res}}^k = \frac{\max(r_{i_{res}}^k)}{r_{i_{res}}^k}$ ,  $\forall k \in \{1, \dots, n\} \setminus \{j\}$  ▷ Sec. 3.5.7.2
18     $s_k^j = \frac{d_{i_{den}}^k}{\sum_{l \neq j} d_{i_{den}}^l} \times \frac{\bar{r}_{i_{res}}^k}{\sum_{l \neq j} \bar{r}_{i_{res}}^l}$ ,  $\forall k \in \{1, \dots, n\} \setminus \{j\}$  ▷ Sec. 3.5.7.2
19     $s_k^j = \frac{s_k^j}{\sum_{l \neq j} s_l^j}$ ,  $\forall k \in \{1, \dots, n\} \setminus \{j\}$ ,  $s_j^j = 0$  ▷ conditional inlier prob.
20     $\tau_{curr}^j = \frac{\sum_{i \in \theta^j} d_i^j}{|\theta^j|}$  ▷ explanation score. Sec. 3.5.7.3
21  end
22   $v = \{k \mid \frac{\tau_{curr}^k - \tau_{prev}^k}{\tau_{prev}^k} \geq \alpha\}$   $\forall k \in \{1, \dots, n\}$  ▷ update index set.
23   $\mathbf{C} = \text{updatePointCorrelation}(\mathbf{C}, v)$  ▷ point correlation. Sec. 3.5.4.
24   $\tau_{prev}^k = \tau_{curr}^k \quad \forall k \in \{1, \dots, n\}$  ▷ update explanation scores.
25 end
26  $\Theta = \bigcup_{k=1}^n v_k^k$  ▷ set of Top-1 preferences of all data points. Sec. 3.5.2.2.
27  $\mathbf{H} = \{\mathbf{h}_i\}_{\forall i \in \Theta}$  ▷ final set of hypotheses,
28  $\mathbf{R} = \{\mathbf{r}_i\}_{\forall i \in \Theta}$ ,  $\mathbf{D} = \{\mathbf{d}_i\}_{\forall i \in \Theta}$  ▷ and their residual and KRD score vectors.

```

---

### 3.5.8 KDGS Algorithm Flow

The KDGS algorithm maintains an index set  $\nu$  that contains the indices of data points for which hypotheses are yet to be generated. It starts with initializing the set  $\nu = \{1, \dots, n\}$  (Algo. 1, line 2), the point correlation matrix (PCM)  $\mathbf{C}$  of size  $n \times n$  with all elements equals to one (Algo. 1, line 2). We assign equal conditional probability to all data points in the conditional inlier probability matrix  $\mathbf{S}$  of size  $n \times n$  with all diagonal elements equals to zero (Algo. 1, line 3). The initial  $\mathbf{C}$  matrix represents that every point pair is equally correlated. Every  $j^{\text{th}}$  column in the  $\mathbf{S}$  matrix represents a discrete probability distribution conditioned on the  $j^{\text{th}}$  data point's potential good hypotheses. The initial  $j^{\text{th}}$  column in the matrix  $\mathbf{S}$  indicates all data points (except  $j^{\text{th}}$  point, which is assigned zero probability to ensure sampling without replacement) are equally likely to be an inlier of the same structure to which  $j^{\text{th}}$  data point belongs.

The  $\mathbf{C}$ ,  $\mathbf{S}$ ,  $\nu$  and  $\eta$  are fed into the routine `generateHyps` (Algo. 2) to generate hypotheses. For each data point  $j \in \nu$ , the routine `generateHyps` generates a hypothesis by sampling the elements of the corresponding minimal sample set  $\mathcal{M}$  from conditional discrete inlier probability distribution derived from  $\mathbf{c}^j$  and  $\mathbf{s}^j$  (Sec. 3.5.7.4). The important steps in Algo. 2 are in lines 3-6, where for each  $j^{\text{th}}$  data point ( $j \in \nu$ ), we deterministically select the data point itself to be the first element of the MSS ( $\mathcal{M}$ ) (Algo. 2, line 3) and the remaining  $\eta - 1$  elements of  $\mathcal{M}$  are sampled from conditional inlier discrete probability distribution (Algo. 1, lines 5-6). Using the data points with indices in the set  $\mathcal{M}$ , we generate a model hypothesis  $\mathbf{h}$  using the function `fitModel` (Algo. 2, line 8), which is a deterministic function and is known a-priori for different model fitting tasks (plane, homography, vanishing point, etc.). We collect the  $|\nu|$  hypotheses in  $\mathbf{H}$  (Algo.2, line 9) and compute the corresponding residual matrix  $\mathbf{R}$  (Algo.2, line 11), and scaled KRD score matrix  $\mathbf{D}$  (Algo.2, line 12).

---

#### Algorithm 2: `generateHyps`

---

```

1 Input:  $\mathbf{C}, \nu, \eta, \mathbf{S}$ , Output:  $\mathbf{H}, \mathbf{R}, \mathbf{D}$ 
2 foreach  $j \in \nu$  do
3    $\mathcal{M}_1 = j$  ▷ deterministic selection of the first element of MSS.
4   for  $k = 2$  to  $\eta$  do
5     construct  $\mathcal{P}_{k-1}$  using  $\mathbf{c}^j$  and  $\mathbf{s}^j$  ▷ Sec. 3.5.7.4.
6      $\mathcal{M}_k \sim \mathcal{P}_{k-1}$  ▷ conditional sampling. Sec. 3.5.7.4.
7   end
8    $\mathbf{h} = \text{fitModel}(\mathcal{M})$ 
9    $\mathbf{H} = \{\mathbf{H} \cup \mathbf{h}\}$ 
10 end
11  $\mathbf{R} = \text{computeResiduals}(\mathbf{H})$ 
12  $\mathbf{D} = \text{computeScaledKRDScores}(\mathbf{R})$  ▷ scaled KRD scores. Sec. 3.5.2.1.

```

---

We combine the outputs of the routine `generateHyps`, the generated hypotheses  $\mathbf{H}$  and the corresponding residual  $\mathbf{R}$  and KRD scores  $\mathbf{D}$  in the current iteration with our previous set (Algo. 1, line 6). With a slight abuse of notation we use set union even with the matrices, only to emphasize uniqueness of rows after the update. The KRD based point preferences  $\mathbf{V}$  are computed at line 12 of Algo. 1 following the process described in (Sec.

3.5.2.2). For each  $j^{\text{th}}$  data point we compute its potential good hypotheses, conditional discrete inlier probability distribution  $\mathbf{s}^j$ , and explanation score  $\tau^j$ . The *explanation score* of all data points in the current iteration  $\tau_{curr}$  is compared with the previous iteration  $\tau_{prev}$ , if the difference between  $\tau_{curr}$  and  $\tau_{prev}$  is insignificant, we exclude those data points from the set  $\nu$ . For the remaining points in the set  $\nu$  we update the corresponding pairwise correlation values in the point correlation matrix  $\mathbf{C}$ . The whole process is repeated until the set  $\nu$  is empty. Finally, we only retain the unique set of hypotheses which are in the top-1 preference of all points and the corresponding residual and density matrices (Algo. 1, lines 26-28).

### 3.5.8.1 Termination Analysis of KDGS

We analyze the behaviour of *explanation score*  $\tau_{curr}$  and  $\nu$  with every iteration of KDGS algorithm (the while loop of Algo. 1). We use two sequences each of motion segmentation (top 2 rows) and planar segmentation (bottom 2 rows) of AdelaideRMF dataset [161] in Fig. 3.7.

The cardinality of set  $\nu$  w.r.t. iterations of the while loop of Algo. 1 is shown in Fig. 3.7 (column 2). It can be seen, the cardinality of the  $\nu$  decreases with every iteration of the while loop of Algo. 1 and reaches to zero. Simultaneously, we can see,  $\tau_{curr}$  of data points of each structure (e.g. st-1, st-2, ...), also saturates at the same iteration when KDGS terminates. For better illustration, we plotted the mean of the explanation scores  $t_{curr}$  of the data points belonging to the same structure. Another important observation is, along with explanation scores ( $\tau_{curr}$ ), the number of good hypotheses generated for each structure also tends to saturate when the algorithm approaches the termination.

While it is difficult to *guarantee* that termination of Algo. 1 with an *upper bound on the number of generated hypotheses*, the strong empirical evidence is shown in Fig. 3.7 indicates the KDGS terminates after few iterations and generates hypotheses for all structures. In Sec. 3.6, we compare KDGS with other competitive guided sampling algorithms.

### 3.5.9 KR Based Inlier Noise Scale Estimation

We get a set ( $\mathbf{H}$ ) of model hypotheses from the KDGS algorithm. The next step is to estimate the inlier noise scale for each model hypothesis in the set  $\mathbf{H}$ . Instead of directly estimating the scale of inlier noise, we first equivalently estimate the fraction of inliers for each of the generated hypotheses in  $\mathbf{H}$ . Given a model hypothesis, using its preference set  $\mathbf{q}$ , we reduce the inlier fraction estimation to the problem of finding an index  $\mathcal{B} \in \{1, \dots, n\}$  that partitions  $\mathbf{q}$  into an inlier subset  $[q_i^1, q_i^2, \dots, q_i^{\mathcal{B}}]$  and an outlier subset  $[q_i^{\mathcal{B}+1}, \dots, q_i^{n-1}, q_i^n]$ . The index  $\mathcal{B}$  then marks the inlier/outlier boundary for the corresponding model hypothesis and the corresponding fraction of inliers is simply  $\frac{\mathcal{B}}{n}$ .

We note that the boundary index will always lie beyond the index of maximum density, i.e.,  $\mathcal{B} \geq k_1$ , where  $k_1 = \arg \max_j d_i^j$ . We ignore extreme outliers by considering boundary candidates that have residuals smaller than a certain (adaptive) threshold. Let  $k_2$  be the index having the largest residual smaller than  $b\rho_i^{2\eta}$ , where  $b$  is a large constant (in our case, always 50), then we only consider boundary candidate indices in the set  $\{k_1, \dots, k_2\}$ .

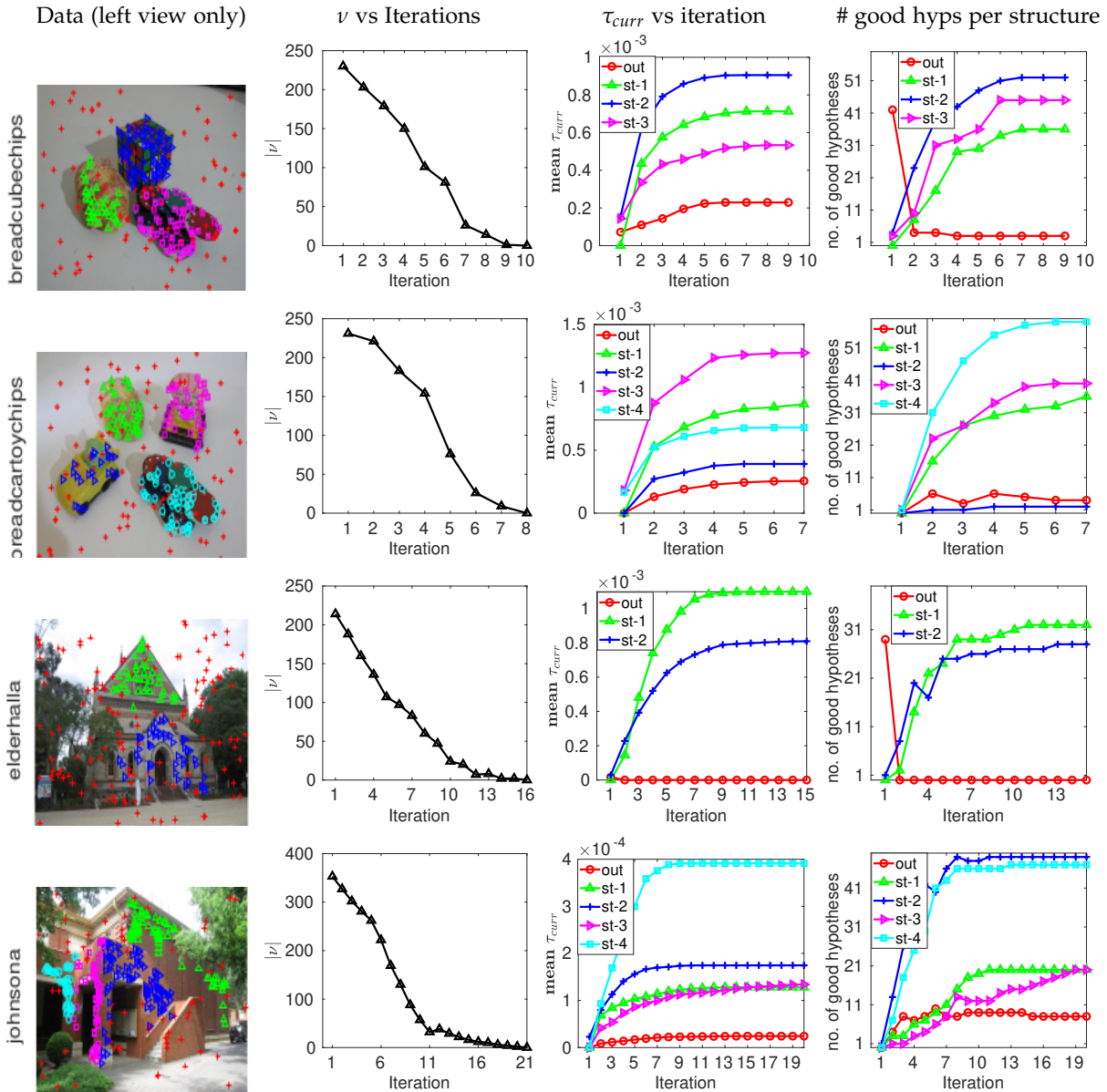


Figure 3.7: Termination analysis of Kernel Density Guided Sampling (KDGS). Refer text in Sec. 3.5.8.1 for detail. We show the ground truth inliers in the one of the two views.

Since inliers are expected to have a significantly higher residual density than outliers, a reliable property of a candidate boundary point is one with a large density difference with respect to the maximum density,  $\rho_i^{k_1}$ . While considering moderately large outliers and inliers, the dispersion of residuals computed over a local neighborhood window  $w_j$  is expected to be large at the inlier/outlier boundary. Based on these intuitions, we define a confidence score using two factors: the normalized local dispersion of residual  $\bar{\zeta}_i^j$  and the normalized density difference  $\bar{d}_i^j$  (Algo. 3, line 4). The inlier fraction  $\hat{f}_i$  of  $i^{th}$  hypothesis is

computed as shown in Algo. 3, lines 5 and 6 respectively.

---

**Algorithm 3:** estimateFraction

---

**Input :**  $\mathbf{d}_i, \rho_i, \eta, b$

**Output:**  $\hat{f}_i$

- 1  $\zeta_i^j \leftarrow \text{std}(\rho_i^{\{w_j\}}), \forall j \in \{1, \dots, n\}$
  - 2  $k_1 \leftarrow \arg \max_j d_i^j$
  - 3  $k_2 \leftarrow \arg \max_j (\rho_i^j \leq b \rho_i^{2\eta})$
  - 4  $\bar{\zeta}_i^j \leftarrow \frac{\zeta_i^j}{\sum_{a=k_1}^{a=k_2} \zeta_i^a}, \bar{d}_i^j \leftarrow \frac{|d_i^j - d_i^{k_1}|}{\sum_{a=k_1}^{a=k_2} |d_i^a - d_i^{k_1}|} \quad \forall j \in \{k_1, \dots, k_2\}$
  - 5  $\mathcal{B}_i \leftarrow \arg \max_j (\bar{\zeta}_i^j \times \bar{d}_i^j) \quad \forall j \in \{k_1, \dots, k_2\}$
  - 6  $\hat{f}_i \leftarrow \frac{\mathcal{B}_i}{n}$  ▷ estimated inlier fraction.
- 

For each  $i^{\text{th}}$  hypothesis, we input its *Kernel Residual Density* (from Sec. 3.5.2.1)  $\mathbf{d}_i$ , sorted residual vector  $\rho_i$  (from Sec. 3.5.1) and size of MSS  $\eta$  to the fraction estimation algorithm *estimateFraction* and record the estimated fraction in  $\hat{f}_i$ . The *estimateFraction* algorithm estimates the fraction by finding an inlier/outliers boundary using density disparity, and dispersion of residuals. From estimated fraction we can compute the number of estimated inliers ( $t_i$ ) as  $t_i = \lfloor \hat{f}_i n \rfloor$ . We use the sorted residual vector  $\rho_i$  and the estimated number of inliers  $t_i$  of  $i^{\text{th}}$  hypothesis to estimate its corresponding inlier noise scale  $\hat{\sigma}_i$  as below.

$$\hat{\sigma}_i = \sqrt{\frac{\sum_{j=1}^{t_i} (\rho_i^j - \text{mean}(\rho_i^{1:t_i}))^2}{t_i - 1}} \quad (3.16)$$

*Note:* since we estimate noise scale for each hypothesis independently, we can efficiently handle the cases where different structures may have different noise scale.

### 3.5.10 Greedy Method/Optimization Based Model Selection

In this section, we propose two variants of model selection: *greedy* and a *quadratic program based model selection*. Before describing model selection algorithms, we first detail a few preliminaries.

For each hypothesis  $\mathbf{h}_i$  we obtain its *estimated* inlier set as  $\mathcal{I}_i = \{q_i^1, q_i^2, \dots, q_i^{t_i}\}$ , where  $t_i$  is the number of estimated inliers, and  $\mathbf{q}_i = \{q_i^1, q_i^2, \dots, q_i^n\}$  is its hypothesis preference set (Sec. 3.5.1). Due to the nature of the hypothesis generation process, there may be multiple good model hypotheses that can explain the same inlier structure. The goal of model selection is to retain the most representative model hypothesis and discard the redundant ones. To identify the best model, we need a measure to quantify the goodness of a model hypothesis. Additionally, since the number of structures is *not known* a priori, therefore, we also need a measure to enforce diversity in the selected models. For the latter, we estimate the pairwise correlation between hypotheses by computing the Spearman-Footrule ( $\mathcal{SF}$ ) distance [160, 42] between their estimated *inlier only* preference lists.

### 3.5.10.1 Hypothesis Correlation using Spearman-Footrule

For each hypothesis pair  $\mathbf{h}_i$  and  $\mathbf{h}_k$ , let their respective top- $t$  inlier only preference list are denoted by  $\bar{\mathbf{q}}_i = [q_i^1, q_i^2, \dots, q_i^t]$  and  $\bar{\mathbf{q}}_k = [q_k^1, q_k^2, \dots, q_k^t]$ , where,  $t = \min(t_i, t_k)$  (i.e. minimum of the number of estimated inliers). The Spearman-Footrule distance is computed using (3.17), where  $Y(\bar{\mathbf{q}}_i)$  denotes the data points with indexes in  $\bar{\mathbf{q}}_i$  and  $j^{\bar{\mathbf{q}}_i}$  denotes the position of the data point ( $j$ ) in the preference list  $\bar{\mathbf{q}}_i$ . We use  $j+1$  for  $j^{\bar{\mathbf{q}}_i}$  if  $j \notin Y(\bar{\mathbf{q}}_i)$ . Variables for  $\bar{\mathbf{q}}_k$  are similarly defined.

$$\mathcal{SF}(\bar{\mathbf{q}}_i, \bar{\mathbf{q}}_k) = \sum_{j \in Y(\bar{\mathbf{q}}_i) \cup Y(\bar{\mathbf{q}}_k)} |j^{\bar{\mathbf{q}}_i} - j^{\bar{\mathbf{q}}_k}| \quad (3.17)$$

$$z_i^k = 1 - \frac{1}{t \times (t+1)} \mathcal{SF}(\bar{\mathbf{q}}_i, \bar{\mathbf{q}}_k) \quad (3.18)$$

We compute pairwise hypothesis correlation between  $\mathbf{h}_i$  and  $\mathbf{h}_k$  using Eq. 3.18. The pairwise hypothesis correlation range between 0 and 1 i.e.  $z_i^k \in [0, 1]$ . A perfect correlation of  $z_i^k = 1$  indicates that both  $\mathbf{h}_i$  and  $\mathbf{h}_k$  have identical inlier only preference lists, while  $z_i^k = 0$  indicates completely dissimilar. We construct a binary similarity matrix  $\mathbf{B}$  by thresholding  $z_i^k \geq \delta$ . A pair of hypotheses is said to be similar, i.e.  $b_i^k = 1$ , if  $z_i^k \geq \delta$ , else dissimilar i.e.  $b_i^k = 0$ .

### 3.5.10.2 Model Hypothesis Goodness Measure

For each model hypothesis  $\mathbf{h}_i$  we measure its goodness  $g_i$ . We define it as the ratio of median density of estimated inliers and top- $\beta$  closest outliers weighted by inverse of estimated inlier noise scale  $\hat{\sigma}_i$ . The goodness score  $g_i$  is computed as shown below in Eq. 3.19, where  $\gamma_i = [t_i + 1, \dots, t_i + \beta]$  (see Sec.3.5.9 for  $t_i$ ) are the indices of top- $\beta$  closest outliers in the residual space.

$$g_i = \frac{\text{median}(d_i^{1:t_i})}{\text{median}(d_i^{\gamma_i})} \times \frac{1}{\hat{\sigma}_i} \quad (3.19)$$

As we have seen in Sec. 3.5.2, for a good hypothesis the density in the inlier region is always greater than in the outlier region. The first term in Eq. 3.19 measures the relative difference between density of estimated inliers and closest outliers, larger the difference better is the hypothesis. The second term captures the compactness of the structure, smaller is the inlier noise scale denser is the structure. A high goodness score of a hypothesis tells the structure it represents through its estimated inliers is dense and has high disparity between density of estimated inlier and outliers.

We use goodness scores  $\mathbf{g} = [g_1, \dots, g_m]$  of all the model hypotheses in our model selection algorithms, which we describe in the next sections.

### 3.5.10.3 Greedy Model Selection (GMS)

GMS aims to iteratively select the model hypothesis in a greedy fashion using goodness score as a selection criterion, and enforce the diversity in the selected models using hypothesis correlation described in the Sec. 3.5.10.1. The complete greedy model selection algorithm is explained in Algo. 4.



**Algorithm 4:** Greedy Model Selection (GMS)

---

```

1 Input:  $\mathbf{g}, \mathbf{B}$ , Output:  $\vartheta$ 
2 Initialization:  $\vartheta \leftarrow \emptyset$ 
3  $\ell \leftarrow \{1, \dots, m\}$  ▷ index set of all generated hypotheses in  $\mathbf{H}$ .
4 while  $\ell \neq \emptyset$  do
5    $k \leftarrow \operatorname{argmax}_i g_i, \quad \forall i \in \ell$  ▷ select model hypothesis with index  $k$ .
6    $\vartheta \leftarrow \{\vartheta \cup \kappa\}$ 
7    $\varrho \leftarrow \{i \mid b_k^i = 1\}, \quad \forall i \in \ell$  ▷ identify hypotheses similar to  $k$ .
8    $\ell \leftarrow \{\ell \setminus \varrho\}$  ▷ remove similar hypotheses from  $\ell$ .
9 end

```

---

**GMS Algorithm.** The GMS algorithm starts with initializing the index set  $\ell = \{1, \dots, m\}$ , which, contains the indices of all hypotheses in  $\mathbf{H}$ . The Algo.4 takes model goodness scores  $\mathbf{g} = [g_1, g_2, \dots, g_m]$  and the binary similarity matrix  $\mathbf{B}$  (Sec. 3.5.10.1). The algorithm begins with selecting the hypothesis with maximum goodness score (Algo. 4, line 5) (say the hypothesis with index  $k$  (i.e.  $\mathbf{h}_k$ ) has the maximum goodness score). Next, it identifies the hypotheses similar to  $k^{\text{th}}$  hypothesis in the set  $\varrho$  and remove them from the set  $\ell$ . The removed hypotheses are high likely the representative hypotheses of the structure already explained by  $k^{\text{th}}$  hypothesis. The process is repeated until the set  $\ell$  is empty. We get the final set of fitted models in  $\vartheta$  (Algo. 4, line 6).

**3.5.10.4 Optimization Based Model Selection**

We formulate the model selection as a constrained quadratic program, where our objective is to maximize the total goodness score, simultaneously enforce the diversity in the selected models. The objective function and the constraint of the Optimization Based Model Selection is shown in Eq. 3.20, where,  $\mathbf{g} = [g_1, g_2, \dots, g_m]$  contains the model goodness score of all the hypotheses in  $\mathbf{H}$ ,  $\lambda$  is the regularization constant and  $\mathbf{Q}$  is a symmetric matrix. The  $\mathbf{Q}$  matrix enforces the diversity in the solutions and is derived from the symmetric hypothesis correlation matrix  $\mathbf{Z}$  (Sec. 3.5.10.1), the diagonal penalty matrix  $\mathbf{P}$  (construction discussed in Sec. 3.5.10.5) and the model goodness scores  $\mathbf{g}$ .

$$\begin{aligned} \max_{\mathbf{y}} \quad & \mathbf{g}^T \mathbf{y} - \lambda \mathbf{y}^T \mathbf{Q} \mathbf{y} \\ \text{s.t.} \quad & \mathbf{y} \in [0, 1]^{m \times 1} \end{aligned} \tag{3.20}$$

$$\mathbf{Q} = \max(\mathbf{g}) \times \mathbf{Z} + \mathbf{P} \tag{3.21}$$

The solution of the quadratic program would be a  $m$  dimensional vector  $\mathbf{y} \in [0, 1]^{m \times 1}$ , where  $m$  is the number of hypotheses in  $\mathbf{H}$ . We use *trust region reflective* algorithm [29, 27] to solve the quadratic program in Eq. 3.20. The trust region is defined by the linear bounds  $0 \leq y_i \leq 1, \forall i \in \{1, \dots, m\}$ . Specifically, we use the *quadprog* solver of Matlab<sup>1</sup> to apply *trust region reflective* algorithm. The initial point  $\mathbf{y}_0$  for the optimization is set to  $\mathbf{y}_0 = \mathbf{0.5}^{m \times 1}$ , which is a  $m$  dimensional vector with all entries equal to 0.5. The final set of

<sup>1</sup><https://in.mathworks.com/help/optim/ug/quadprog.html>

fitted models  $\vartheta$  are obtained as  $\vartheta = \{i \mid y_i \geq th_y\}$ , where,  $th_y$  is the threshold we use to hard select the final set of hypotheses. We set  $th_y = 1e-3$  for all our experiments.

Next, we describe the process of constructing the diagonal penalty matrix  $\mathbf{P}$ .

### 3.5.10.5 Diagonal Penalty Matrix

We adopt a tree traversal based construction of diagonal penalty matrix similar to the strategy proposed in [178], however for the construction of trees we use metrics derived from KRD scores and estimated inlier noise scale. The nodes in the tree are hypotheses in  $\mathbf{H}$ . We first iteratively construct trees by connecting edges between the nodes. A directed edge between hypotheses (nodes)  $h_i$  to  $h_k$  indicates  $g_k \geq g_i$  and  $z_k^i > 0.5$ . The detail steps for constructing a diagonal penalty matrix is explained in Algo. 5. The complete process is as follows: for each  $i^{th}$  hypothesis we select the maximum correlated hypothesis (say  $k^{th}$  hypothesis, other than itself) (Algo. 5, line 4). If  $k^{th}$  hypothesis has high goodness score and has at least 0.5 correlation with  $i^{th}$  hypothesis (Algo. 5, line 5), we add an edge from  $h_i$  to  $h_k$  (Algo. 5, line 6), otherwise, we add an edge to itself (Algo. 5, line 7). We repeat this process for all the hypothesis in  $\mathbf{H}$ . A snapshot of the output of Algo. 5, lines 4-7 is shown in Fig. 3.8.

Once the trees are constructed, for each node (equivalently hypothesis  $\mathbf{h}_i$ ), we traverse and find its root node (Algo. 5, line 9) e.g. refer Fig. 3.8, the root node of  $\mathbf{h}_4$ ,  $\mathbf{h}_8$   $\mathbf{h}_6$  and  $\mathbf{h}_{15}$  are  $\mathbf{h}_3$ ,  $\mathbf{h}_3$ ,  $\mathbf{h}_9$  and  $\mathbf{h}_{15}$  respectively. Let the root node of  $h_i$  is indexed by  $i_r$  (Algo. 5, line 10). If  $i \neq i_r$ , we compute the respective diagonal entry  $p_i^i$  in the diagonal penalty matrix  $\mathbf{P}$  as i.e.  $p_i^i = \max(\mathbf{g}) \times (z_{i_r}^i g_i)$  (Algo. 5, line 13). The diagonal matrix is then added to the scaled hypothesis correlation matrix  $\mathbf{Z}$  as shown below in Eq. 3.21 to enforce diversity in the selected models.

---

#### Algorithm 5: Diagonal Penalty Matrix

---

```

1 Input:  $\mathbf{H}, \mathbf{Z}, \mathbf{g}$ , Output:  $\mathbf{P}$ 
2 Initialization:  $\mathbf{P} \leftarrow \mathbf{0}_{n \times n}$ 
3 for  $i \leftarrow 1$  to  $m$  do
4    $k \leftarrow \operatorname{argmax}_j z_j^i$ 
5   if  $k \neq i$ , and  $g_k \geq g_i$  with  $z_k^i \geq 0.5$ 
6     then add an edge from  $\mathbf{h}_i$  to  $\mathbf{h}_k$  ( $\mathbf{h}_i \rightarrow \mathbf{h}_k$ )
7     else add an edge to itself from  $\mathbf{h}_i$  to  $\mathbf{h}_i$  ( $\mathbf{h}_i \rightarrow \mathbf{h}_i$ )
8 end
9 for each node ( $\mathbf{h}_i$ ), perform tree traversal and find its
10 root node  $\mathbf{h}_{i_r}$  indexed by  $i_r$ .
11 for  $i \leftarrow 1$  to  $m$  do
12   if  $i \neq i_r$  then
13      $p_i^i = \max(\mathbf{g}) \times (z_{i_r}^i g_i)$ 
14   end
15 end

```

---

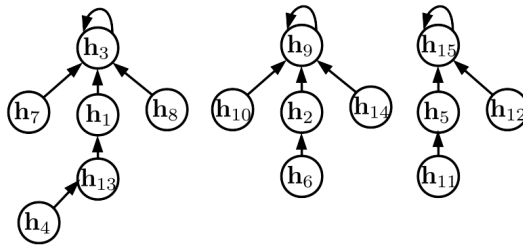


Figure 3.8: If there are a total of 15 hypotheses in  $H$ , a possible output of Algo. 5, lines 4-7 may look like this. Note: This is just for the illustration purpose, the number of trees and hypotheses may vary.

### 3.5.11 Point-to-Model Assignment

The model selection algorithm output the set of indices ( $\vartheta$ ) of our final selected model hypotheses, and their associated inlier sets  $\mathcal{I}_i, i \in \vartheta$ . At this stage, some of the data points may be members of multiple sets  $\mathcal{I}_i$  and  $\mathcal{I}_j$  for  $i, j \in \vartheta$ . This is acceptable for soft partitioning, however, we reassign the points based on the KRD scores to achieve hard partitioning of data points. That is, a point (say  $x^j$ ) can be associated to *only one* inlier set (say  $\mathcal{I}_k$ ), provided  $d_k^j \geq d_i^j \forall i \in \{\vartheta \setminus k\}$ . We refine the inlier sets following above KRD score based point-to-model assignment strategy.

## 3.6 Experimental Analysis

In this section, we evaluate our proposed DGSAC pipeline. We first present an experimental evaluation of our KRD Guided Sampling (KDGS) in Sec. 3.6.1 and then we evaluate the full DGSAC pipeline in Sec. 3.6.2. We present the evaluation of two variants of our full pipeline. One is a guided sampling (KDGS) with greedy model selection (dubbed as DGSAC-G), the other is, guided sampling (KDGS) with optimization based model selection (dubbed as DGSAC-O). We evaluate full DGSAC pipeline on wide variety of applications: planar segmentation, motion segmentation, vanishing point estimation/lines classification, plane fitting to 3D point cloud, line and circle fitting.

In DGSAC, we generate hypotheses by fitting model equation to more than minimal sample subset of the data. Specifically, we estimate homography, fundamental matrix, vanishing point by fitting it to the subset of data with cardinality one more than minimal sample subset. We fit line, plane, and circles to two more than minimal subset of data.

**Datasets:** The datasets we use in this evaluation for the respective applications are as follows:

Planar and Motion Segmentation: We use standard AdelaideRMF [161] dataset. It consists of 19 sequences each for planar and motion segmentation. The dataset provides labeled SIFT[87] point correspondences in two-view and the ground-truth labeling for each point correspondence. We use ground-truth only for the evaluation purpose.

Vanishing Point Estimation: We use the York urban line segment [34] and the Toulouse Vanishing Points [2] data sets. The York Urban and Toulouse Vanishing Point data sets comprise 102 and 110 indoor and outdoor urban scenes. The ground truth classification of lines to the three dominating vanishing directions is provided.

**Plane Fitting to 3D Point Cloud:** We use two real examples *CastelVechio* and *PozzoVegiani* of SAMANTHA [43] data set. Since the ground truth labeling of data points is not available. We show qualitative results for this application.

**Line and Circle Fitting:** We use *Star5* and *Circle5* dataset from [144]. These examples are synthetically generated with Gaussian noise  $\sigma = 0.0075$  and 50% outliers.

*Table 3.1: Qualitative Evaluation of KDGS (Planar Segmentation). #H is the total number of hypotheses generated by each method, #HM(%) is the percentage of good hypotheses satisfying the all inlier MSS criteria, #HI(%) percentage of good hypotheses having at-least 80% overlap of their estimated inliers with the true inliers,  $n$  is the number of point correspondences,  $O(\%)$  is the outlier percentage. The  $\mathcal{T} = [\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_\kappa]$  vector shows the number of true inliers of all  $\kappa$  genuine structures. tim(s) shows the total time taken in seconds. Image shows two-view visual ground-truth inliers with color coded structural membership. Outliers are in red. Best result is in bold and second best is underlined.*




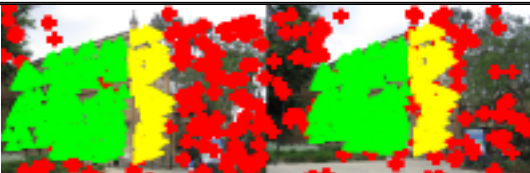
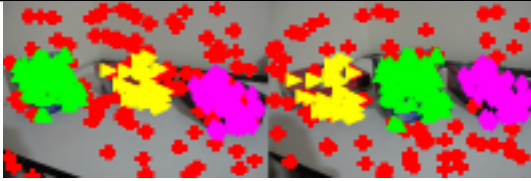

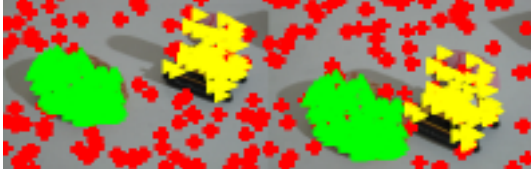
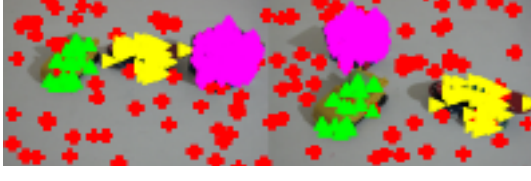
Planar Segmentation					
Data		MGS	ITKSF	DHF	DGS
 elderhalla, $n = 214$ , $O = 60.8\%$ $\mathcal{T} = [ 38, 46 ]$	#H	463	201	61	57
	#HM	18.3	26.8	<u>44.8</u>	<b>96.1</b>
	#HI	16.7	21.3	<u>31.2</u>	<b>100.0</b>
	tim	1.90	1.90	1.90	1.90
 johnsona, $n = 353$ , $O = 21.2\%$ $\mathcal{T} = [ 76, 91, 61, 50 ]$	#H	956	331	68	140
	#HM	34.3	47.4	<u>70.2</u>	<b>84.8</b>
	#HI	31.4	34.6	<u>37.8</u>	<b>92.8</b>
	tim	11.60	11.60	11.60	11.60
 ladysymon, $n = 227$ , $O = 33.5\%$ $\mathcal{T} = [ 102, 49 ]$	#H	579	215	59	37
	#HM	38.3	46.1	<u>58.4</u>	<b>90.8</b>
	#HI	<u>36.5</u>	34.3	24.5	<b>61.6</b>
	tim	2.65	2.65	2.65	2.65
 oldclassicswing, $n = 363$ , $O = 32.2\%$ $\mathcal{T} = [ 181, 65 ]$	#H	939	343	66	32
	#HM	48.8	55.2	<u>74.8</u>	<b>95.7</b>
	#HI	<u>51.9</u>	49.6	50.0	<b>95.2</b>
	tim	11.30	11.30	11.30	11.30

Table 3.2: *Qualitative Evaluation of KDGS (Motion Segmentation)*. #H is the total number of hypotheses generated by each method, #HM(%) is the percentage of good hypotheses satisfying the all inlier MSS criteria, #HI(%) percentage of good hypotheses having at-least 80% overlap of their estimated inliers with the true inliers,  $n$  is the number of point correspondences,  $O(\%)$  is the outlier percentage. The  $\mathcal{T} = [\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_\kappa]$  vector shows the number of true inliers of all  $\kappa$  genuine structures. *tim(s)* shows the total time taken in seconds. Image shows two-view visual ground-truth inliers with color coded structural membership. Outliers are in red. Best result is in bold and second best is underlined.

Motion Segmentation					
Data	MGS	ITKSF	DHF	DGS	
 biscuitbookbox, $n = 258$ , $O = 37.2\%$ $\mathcal{T} = [ 67, 41, 54 ]$	#H	380	213	68	154
	#HM	32.15	57.98	<u>85.23</u>	<b>86.39</b>
	#HI	55.69	59.41	<u>73.79</u>	<b>97.30</b>
	tim	4.38	4.38	4.38	4.38
 breadcubechips, $n = 230$ , $O = 35.2\%$ $\mathcal{T} = [ 34, 57, 58 ]$	#H	258	181	70	137
	#HM	27.26	55.98	<b>86.98</b>	<u>76.68</u>
	#HI	52.40	73.70	<u>89.16</u>	<b>96.24</b>
	tim	2.15	2.15	2.15	2.15
 breadtoy, $n = 278$ , $O = 37.4\%$ $\mathcal{T} = [ 119, 55 ]$	#H	415	237	76	81
	#HM	28.30	51.56	<b>85.66</b>	80.29
	#HI	58.80	67.77	<u>90.56</u>	<b>97.29</b>
	tim	4.57	4.57	4.57	4.57
 carchipscube, $n = 164$ , $O = 36.6\%$ $\mathcal{T} = [ 18, 33, 53 ]$	#H	176	138	57	84
	#HM	23.64	50.37	<u>69.42</u>	<b>70.21</b>
	#HI	50.37	69.12	<u>84.95</u>	<b>92.86</b>
	tim	1.23	1.23	1.23	1.23

### 3.6.1 Experimental Analysis of KDGS

We compare the proposed guided sampling algorithm KDGS with other state-of-the-art methods like DHF [160], Multi-GS [24], ITKSF [160] for which the authors<sup>2</sup> have publicly released the implementations. The competing methods DHF, ITKSF, and Multi-GS, works in a time budget framework and require a user-specified inlier threshold. In contrast,

<sup>2</sup>We thank Hoi Sim Wong for providing the source code of DHF and ITKSF.

our method (KDGS) is a non-time budget, self-terminating, and does not require an inlier-outlier threshold. Since KDGS is an automated guided sampling method, we first run KDGS and record the time taken for each data sequence. For a fair comparison, we run all three competing methods for the same time budget. We evaluate KDGS on planar and motion segmentation tasks using AdelaideRMF [161] dataset.

**Metrics.** The competing methods have defined a good model hypothesis as a hypothesis fitted on an all inlier MSS. However, it may be possible that a hypothesis fit on an all inlier MSS results is a bad hypothesis due to inherent inlier noise scale [135]. Therefore, in addition to all inlier MSS criteria, we define another strong criterion for defining a good model hypothesis: a hypothesis can be called a good hypothesis if its estimated inliers have at-least 80% overlap with the ground truth-inliers.

**Results.** We report the total number of hypotheses generated by the respective methods (#H), the percentage of good hypotheses based on all inliers MSS (#HM(%)), and the percentage of hypothesis satisfying the 80% overlapping criteria (#HI(%)) in Tab. 3.1 and 3.2. We also report the total time taken *tim* (in seconds) by the KDGS algorithm. While we have also reported the total time taken, it may not be a strictly fair comparison as some parts of the competing methods are implemented in the C programming language. DGSAC is fully implemented in Matlab; therefore, further improvement in running time is possible with an optimized implementation.

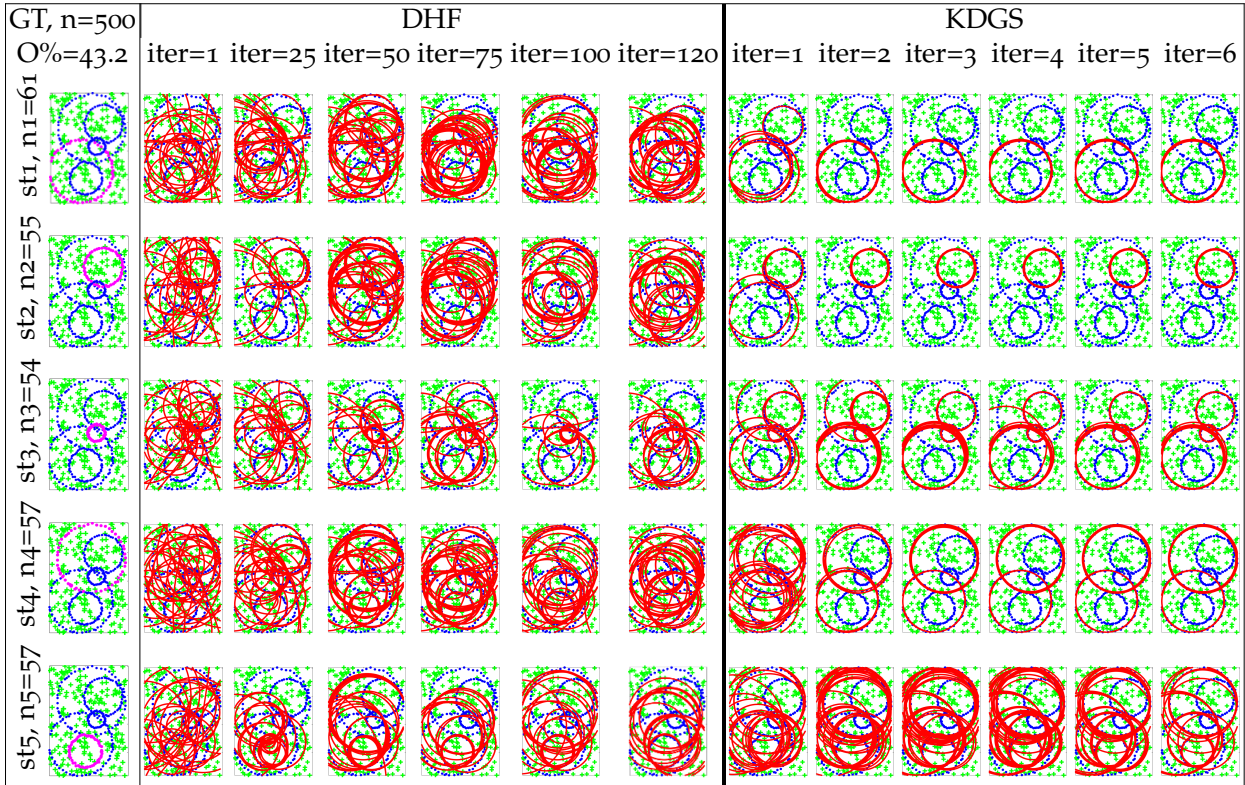
**Analysis of results in Tab. 3.1 and 3.2.** While previous guided sampling algorithms use a guessed time budget for running the guided sampling algorithm, the proposed KDGS self-terminates after generating hypotheses explaining all the data points. From the results in Tab. 3.1 and 3.2, it can be seen that compared to other competing methods, KDGS can generate a high percentage of good hypothesis and self-terminate within a time which is an order of magnitude smaller (in most of the cases) than the usual time budget (*i.e.* 10s). In some cases, *johnsona*, and *oldclassicswing*, the time taken by KDGS is slightly more than 10s. The stopping criteria must correlate with the data's information, *e.g.*, number of structures, no. of data points. It should not be merely a guess. It is impossible to guess a lower bound on the time budget for all possible tasks. The KDGS uses an explanation score to stop the sampling process- a data-driven approach that can adapt according to the underlying fitting task.

Consider *breadtoy* example, where, #HM is smaller than #HI. The Small value of #HM indicates that not all hypothesis fitted on all inlier MSS are good. It happens due to the inherent large noise scale within the true structure, which leads to a hypothesis fitted on all inlier MSS capable of describing less than 80% of true inliers. There #HI is a more robust criterion to decide on a good hypothesis

**DHF vs KDGS.** The closest method to KDGS is the DHF. DHF also follows a per point guided sampling approach. We compare KDGS with DHF using a multiple-circle fitting example on *Circle5* dataset. We show the quality of hypotheses generated by KDGS and DHF w.r.t. the sampling iterations in Tab. 3.3. It is evident, KDGS quickly starts sampling

within true structures and generate a high fraction of good hypotheses.

Table 3.3: Comparison of KDGS with DHF. Both DHF and KDGS focus on generating hypothesis for each data point, the plotted hypotheses (in red) are of the inliers of five genuine structures ( $s_1, \dots, s_5$ ) shown in column 1.  $n$  is the total number of data points,  $O\%$  is the ground-truth (GT) percentage of gross outliers. Ground-truth structure is shown in column 1 (in magenta) and outliers are in green. For better illustration we have plotted hypotheses of each of the five structures separately in the five different rows.



**Outlier rate(%) vs. % of good hypotheses.** We analyze the effect of the outlier rate (%) on the performance of DHF, ITKSF, Multi-GS, and KDGS. We have plotted the #HM and #HI metrics against the outlier rate(%) in Fig. 3.9. We see a drop in the performance *i.e.* percentage of good hypotheses generated by DHF, ITKSF, and Multi-GS with the increase in the outlier rate(%). The KDGS performance does not significantly affect the outlier rate. A similar decrease in the number of good hypotheses generated by DHF, ITKSF, and Multi-GS is observed in [160].

Table 3.4: Required user inputs ( $\epsilon$ = inlier/outlier threshold,  $\kappa$  = no. of structures): ( $\checkmark$  = Required,  $\times$  = Not Required) . In addition to  $\epsilon$ , Prog-X require more user-defined thresholds, details are in [6].

	RPA	Tlink	RCM	DPA	Cov	QP-MF	NMU	Prog-X	DGSAC-G/O
$\epsilon$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\times$
$\kappa$	$\checkmark$	$\checkmark$	$\times$	$\times$	$\checkmark$	$\checkmark$	$\times$	$\times$	$\times$

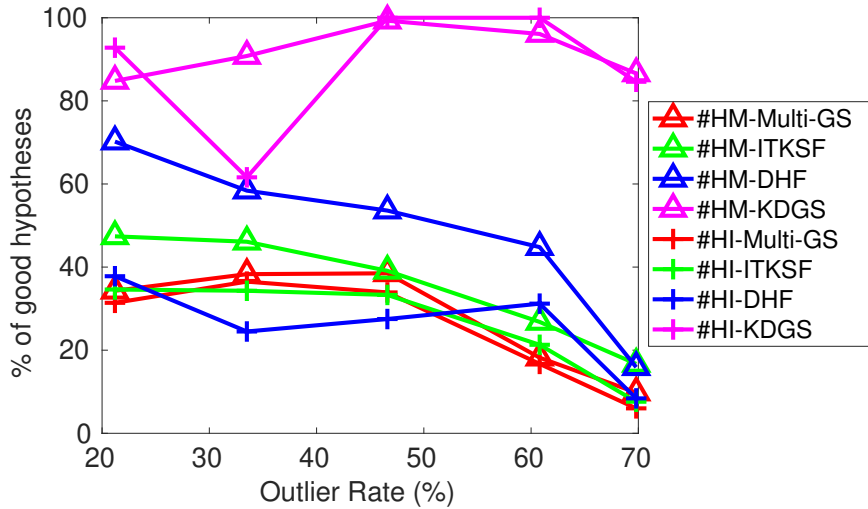


Figure 3.9: Percentage of good hypotheses w.r.t. the outlier rate (%).

### 3.6.2 Evaluation of full DGSAC Pipeline

In this section, we evaluate two variants of our DGSAC pipeline: *DGSAC-G* (KDGS with Greedy Model Selection) *DGSAC-O* (KDGS with Quadratic Program based Model Selection).

**Competing Methods.** We compare our full DGSAC pipeline on variety of tasks with state-of-the-art multi-model fitting methods for which the source code is publicly released by the respective authors, like J-Linkage (Jlink) [144], T-Linkage (Tlink) [95], RPA [93], DPA [141], RCM [110], RansaCov (Cov) [92], NMU [137], QP-MF [178], Prog-X [6], L1-NMF [136]. We follow the guidelines mentioned by the authors in their papers and provide the necessary parameters like *user specified inlier threshold*, *number of models*, or *both*. Our DGSAC-G/O is the only method that does not require an inlier threshold or the number of models. A comparison of competing methods based on the dependency on user inputs is shown in Tab. 3.4, these user inputs are mostly computed from the ground truth.

**Metrics.** We use Classification Accuracy (CA) as an evaluation metric, i.e., the percentage of data points correctly assigned to their respective true structures or gross outliers category. All results are averaged over ten runs. We re-emphasize that, while we have also reported the running time, it is not a strictly fair comparison as the programming language varies across the competing approaches. DGSAC is implemented in Matlab; therefore, further improvement in running time is possible with an optimized implementation.

#### 3.6.2.1 Evaluation on Real Datasets

In this section, we present evaluation results on four multi-model fitting tasks using real datasets: *motion segmentation*, *planar segmentation*, *vanishing point estimation* and *plane fitting* to 3D point cloud.



Table 3.5: *Quantitative Analysis on Motion Segmentation. Classification Accuracy(CA) in (%), Total time taken including both KDGS and model selection in seconds, O(%)= Outliers Percentage,  $\kappa$  = number of true structures,  $\mu$ =mean, med=median. DGSAC\* is the DGSAC version proposed in [139]. Results are divided into two block separated by a dashed line. The methods in the top block require either user specified inlier threshold, number of models or both, while methods in second block does not require inlier threshold or number of models. Best result is in bold and second best is underlined.*

	biscuit	biscuitbook	bisbookbox	boardgame	book	brdcartoychips	breadcube	brdcubechips	breadtoy	breadtoycar	carchipscube	cube	cubebrdtoyboxes	cubebechips	cube toy	dinobooks	game	gamebiscuit	toybecar	CA(%)	Time(s)		
$n$	319	341	258	266	185	231	233	230	278	164	164	295	314	277	239	339	230	324	198	$\mu$	$med$	$\mu$	$med$
O(%)	57.2	47.5	37.2	42.5	21.5	35.2	32.2	35.2	37.4	34.2	36.6	69.5	28.0	51.6	41.4	44.5	73.5	51.5	36.4	$\mu$	$med$	$\mu$	$med$
$\kappa$	1	2	3	3	1	4	2	3	2	3	3	1	4	2	2	3	1	2	3				
Tlink	83.1	<u>97.8</u>	88.8	<u>83.7</u>	82.6	80.5	85.6	82.0	96.8	84.7	88.0	46.3	80.2	95.1	78.8	78.6	77.6	70.6	70.7	81.7	82.6	<u>12.8</u>	<u>11.7</u>
RCM	95.2	92.5	83.7	78.5	94.0	78.8	87.3	83.2	78.4	83.1	78.9	87.9	81.6	90.3	89.6	72.3	90.8	85.4	83.5	85.0	83.7	<b>04.6</b>	<b>03.8</b>
RPA	<b>98.4</b>	96.4	<u>95.8</u>	<b>87.5</b>	<u>97.5</u>	<u>91.7</u>	<u>96.0</u>	<u>95.6</u>	<u>97.2</u>	<b>92.2</b>	<u>94.3</u>	<u>97.2</u>	<b>93.2</b>	<u>96.5</u>	<u>96.3</u>	<b>84.8</b>	95.9	<b>96.9</b>	<b>91.7</b>	<u>94.5</u>	<u>95.9</u>	39.3	38.8
DPA	82.1	97.2	95.1	<u>83.7</u>	90.2	91.6	94.1	94.6	90.6	88.7	86.3	96.9	87.3	92.9	93.6	84.2	<u>97.5</u>	90.9	85.6	90.6	90.9	50.3	46.8
Cov	<b>98.4</b>	97.6	94.0	77.8	97.2	87.3	95.9	88.6	82.4	<u>89.2</u>	88.7	97.1	<u>90.7</u>	93.6	<u>95.5</u>	68.7	92.4	95.5	82.1	90.1	92.4	54.7	47.3
NMU	<u>97.6</u>	<b>98.8</b>	<b>98.1</b>	82.8	<b>100</b>	<b>94.9</b>	<b>97.1</b>	<b>97.4</b>	<b>97.9</b>	<b>92.2</b>	<b>97.6</b>	<b>98.0</b>	87.2	<b>98.6</b>	<b>98.0</b>	<u>84.4</u>	<b>98.7</b>	92.1	<u>91.5</u>	<b>94.9</b>	<b>97.6</b>	399	399
QP-MF	55.8	52.5	62.5	64.4	56.2	65.4	68.2	64.8	63.2	66.3	67.9	67.9	73.1	60.9	60.2	56.9	73.0	60.7	66.5	63.5	64.4	20.3	20.2
DGSAC*	<b>98.2</b>	<b>98.6</b>	<u>97.6</u>	82.6	<u>99.0</u>	<u>87.9</u>	<b>97.7</b>	93.0	90.7	89.5	85.5	<b>96.8</b>	88.6	97.4	<u>97.3</u>	83.9	95.0	<u>98.2</u>	<u>90.4</u>	<u>93.1</u>	<u>95.0</u>	24.2	20.2
DGSAC-G	<u>98.1</u>	<u>98.5</u>	97.3	<b>89.5</b>	<b>99.3</b>	<b>89.0</b>	<u>96.7</u>	<u>97.7</u>	<u>96.1</u>	89.9	<b>88.9</b>	95.9	91.9	97.3	<b>98.03</b>	<b>86.5</b>	97.5	<b>99.0</b>	<b>91.9</b>	<b>94.7</b>	<b>96.7</b>	<u>5.7</u>	<u>5.4</u>
DGSAC-O	88.3	94.8	<b>98.1</b>	<u>83.3</u>	89.3	85.3	89.4	<b>97.8</b>	<b>97.7</b>	<b>93.3</b>	<u>86.6</u>	88.4	<b>94.0</b>	<b>97.4</b>	96.9	<u>85.4</u>	<b>98.7</b>	93.8	89.9	92.1	93.3	<b>5.3</b>	<b>4.9</b>

**Multiple Motion and Planar Segmentation.** We use AdelaideRMF [161] to evaluate DGSAC-G/O on motion and planar segmentation tasks. The quantitative results are reported in Tab. 3.5. NMU achieves the highest accuracy for both motion and planar segmentation task, but it takes *highest running time* and *requires user-specified inlier threshold*. Our DGSAC-G achieves the next best accuracy, lagging by a margin of  $< 1\%$ , and DGSAC-O gives competitive results without any user input (inlier threshold and the number of structures (refer Tab. 3.4) at all. Moreover, in terms of average time to run, DGSAC-G and DGSAC-O are nearly  $68\times$  and  $75\times$  faster than the NMU in motion segmentation  $21\times$  faster in planar segmentation, with the median run times being even better. We also compare with recently proposed optimization-based method Prog-X and another global optimal quadratic program based method QP-MF [178]. The Prog-X require inlier-threshold along with other user inputs [6], while QP-MF require both *user specified inlier threshold* and *number of models* as an inputs. Our optimization DGSAC-O outperforms both Prog-X<sup>3</sup> and QP-MF with a significant margin, without any dependency on the user-specified inputs. We report some sample qualitative results of both motion and planar segmentation tasks in Fig. 3.10 where point membership is color-coded.

**Multiple Vanishing Point Estimation.** We use the York urban line segment [34] and the Toulouse Vanishing Points [2] dataset to evaluate DGSAC-G/O. We compare DGSAC with

<sup>3</sup><https://github.com/danini/progressive-x>. Only planar segmentation implementation is publicly available.

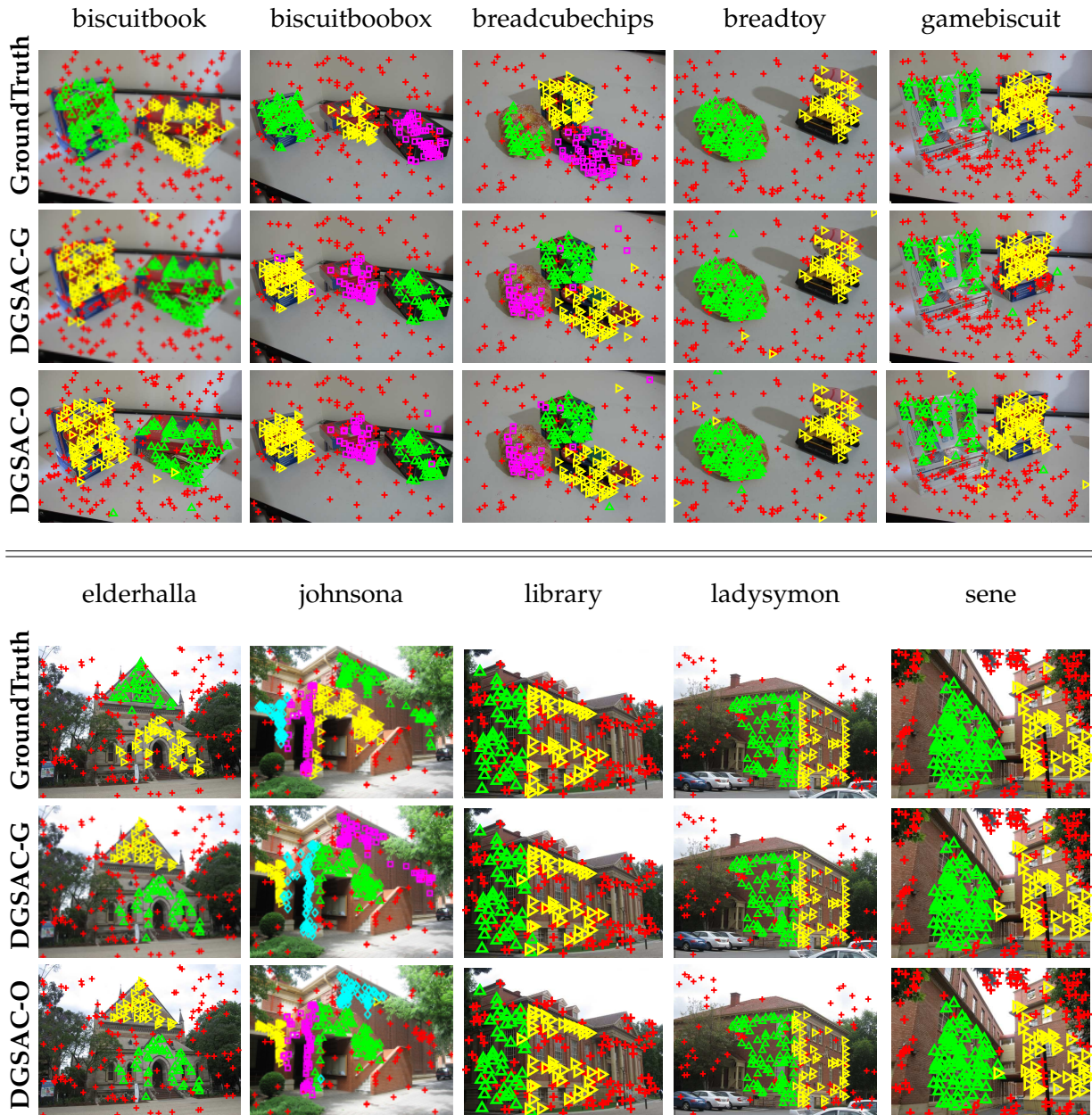


Figure 3.10: *Some Qualitative Results of Motion and Planar Segmentation. Qualitative results are shown for some examples from AdelaideRMF [161] dataset. Motion segmentation: Top-3 rows, Planar Segmentation: Bottom-3 rows. Point membership is color coded. Gross outliers are in red. We have shown only the view 1 of the two views.*

the RANSAC like state-of-the-art vanishing point estimation methods. The quantitative results are reported in Tab. 3.7. DGSAC-G/O outperforms in both the data sets. Sample qualitative results are reported in Fig. 3.11, where point membership is color-coded, *i.e.* lines with the same color belong to the same vanishing point direction.

**Multiple Plane Fitting to 3D Point Cloud.** We use two real examples *CastelVechio* and *PozzoVeggiani* from SAMANTHA [43] dataset to evaluate DGSAC-G/O. The ground-truth labeling is not provided with the data set. Therefore, we report qualitative results in Fig. 3.12, where point membership is color-coded. Only J-Linkage and DGSAC-G/O recover

Table 3.6: Quantitative Analysis on Planar Segmentation. Notations are same as of table 3.5.

	barrsmith	bonhall	bonython	elderhalla	elderhallb	hartley	johnsona	johnsonb	ladysymon	library	napiera	napierb	neem	nese	oldclassics	physics	sene	unihouse	unionhouse	CA(%)		Time(s)	
$n$	235	948	193	214	245	315	353	624	227	212	292	237	230	241	363	103	236	1784	321	$\mu$	$med$	$\mu$	$med$
O(%)	68.9	06.2	73.7	60.7	47.8	61.6	20.9	12.0	32.5	55.3	62.9	39.5	36.5	33.5	32.5	45.3	47.2	16.6	76.5				
$\kappa$	2	6	1	2	3	2	4	7	2	2	2	3	3	2	2	1	2	5	1				
<b>Tlink</b>	57.9	60.4	64.3	69.5	57.8	71.6	57.8	70.7	77.7	82.5	81.3	67.7	53.0	53.7	73.8	68.5	84.3	71.9	77.3	69.0	70.7	492	81.3
<b>RCM</b>	84.8	<u>81.7</u>	87.3	75.2	71.5	77.4	83.0	<u>79.4</u>	75.3	77.0	70.7	74.3	71.9	77.6	92.5	54.5	71.7	<u>97.0</u>	90.1	78.6	77.4	<u>5.3</u>	<u>3.4</u>
<b>RPA</b>	62.9	52.9	84.3	<b>99.1</b>	82.0	81.4	91.1	66.8	79.2	63.5	73.3	75.1	78.5	<b>99.2</b>	76.7	<b>100</b>	99.4	88.0	76.1	80.5	79.2	967	247
<b>DPA</b>	<b>97.7</b>	78.0	96.6	96.2	<u>85.9</u>	<u>96.9</u>	87.1	74.4	90.5	95.2	80.6	<b>83.6</b>	<u>80.2</u>	97.4	96.3	98.4	<b>99.8</b>	93.2	98.3	<u>90.9</u>	<u>95.2</u>	37.7	30.1
<b>Cov</b>	70.7	68.6	<b>99.7</b>	77.9	82.8	91.8	86.1	65.2	<u>93.8</u>	92.9	86.6	74.1	72.6	90.8	79.2	<u>99.5</u>	80.4	91.2	<b>99.5</b>	84.2	82.8	145	53.2
<b>NMU</b>	<u>89.6</u>	<b>84.3</b>	<u>98.5</u>	<u>98.1</u>	<b>86.7</b>	<b>98.4</b>	90.6	75.0	<b>96.2</b>	<b>98.1</b>	<b>94.7</b>	78.4	<b>95.9</b>	<u>97.6</u>	<b>98.4</b>	79.3	<u>99.6</u>	94.7	<u>99.2</u>	<b>92.3</b>	<b>96.0</b>	499	298
<b>Prog-X</b>	88.4	74.4	98.3	78.9	80.9	96.9	<b>91.5</b>	<b>82.9</b>	<b>96.2</b>	<u>97.3</u>	87.1	79.2	74.0	96.4	<u>97.7</u>	65.0	97.7	<b>97.4</b>	98.7	88.4	88.7	<b>1.53</b>	<b>1.48</b>
<b>QP-MF</b>	63.9	71.6	73.7	72.9	82.4	55.9	62.2	61.0	60.3	55.4	54.0	73.8	78.8	66.5	77.8	54.7	79.6	80.4	76.5	68.5	71.6	25.2	20.1
<b>DGSAC*</b>	<u>69.6</u>	<u>73.0</u>	<u>98.2</u>	<u>96.8</u>	88.3	<b>97.8</b>	<b>94.9</b>	77.5	<u>91.9</u>	94.2	<u>92.8</u>	<u>82.6</u>	<b>90.6</b>	<u>99.2</u>	<b>94.2</b>	<u>99.4</u>	<u>98.6</u>	<b>92.9</b>	<u>97.1</u>	91.0	<u>94.2</u>	115	23.2
<b>DGSAC-G</b>	<b>89.4</b>	<b>73.4</b>	<b>99.0</b>	<b>99.1</b>	87.6	97.2	<u>94.3</u>	<u>77.6</u>	<b>96.5</b>	<u>97.7</u>	<b>93.8</b>	<b>83.5</b>	<b>88.9</b>	<b>99.6</b>	<b>94.2</b>	<b>100</b>	<b>99.2</b>	<u>92.7</u>	<b>98.8</b>	<b>92.8</b>	<b>94.3</b>	<b>23.8</b>	<u>3.5</u>
<b>DGSAC-O</b>	<b>89.4</b>	72.8	<u>98.4</u>	<b>99.1</b>	<u>87.6</u>	<u>97.5</u>	<u>94.3</u>	<u>77.9</u>	87.4	<b>98.6</b>	89.3	<b>83.5</b>	80.4	<b>99.6</b>	<u>91.8</u>	<b>100</b>	<b>99.2</b>	92.1	<b>98.8</b>	<u>91.5</u>	<u>92.1</u>	<u>24.9</u>	3.4

Table 3.7: Qualitative Evaluation on VP Estimation. Notations are same as in table 3.5.

	York Dataset				Toulouse Dataset			
	CA(%)		Time(s)		CA(%)		Time(s)	
	$\mu$	$med$	$\mu$	$med$	$\mu$	$med$	$\mu$	$med$
<b>RPA</b>	95.4	97.9	04.4	02.4	55.3	54.6	00.8	00.72
<b>Cov</b>	95.6	97.4	01.24	<b>00.26</b>	51.8	50.0	00.04	00.07
<b>L1-NMF</b>	94.1	96.7	<b>00.71</b>	00.31	74.1	75.0	00.07	00.54
<b>DGSAC-G</b>	<b>96.0</b>	<b>98.0</b>	01.29	<u>01.25</u>	<b>92.1</b>	<b>95.9</b>	<b>00.03</b>	<b>00.03</b>
<b>DGSAC-O</b>	<u>95.8</u>	<u>97.9</u>	01.30	01.26	<u>91.9</u>	<u>95.6</u>	00.05	<u>00.05</u>

planes correctly.

### 3.6.2.2 Evaluation on Synthetic Datasets

In this section, we present evaluation results on multiple *line* and *circle* fitting tasks using standard synthetic datasets proposed in [144].

**Multiple Line Fitting.** We use *Star5* dataset from [144]. The qualitative and quantitative results are reported in Fig. 3.13. While all the competing methods are successfully able to recover all the five structures. Our DGSAC-G and DGSAC-O classify 96% of the total data points to their respective classes *i.e.* their true structures or gross outliers. Both DGSAC-G and DGSAC-O output the same set of final hypotheses using greedy model selection and optimization based model selection algorithm, hence the same CA(%).

**Multiple Circle Fitting.** We use *Circle5* dataset from [144]. The qualitative and quantitative results are reported in Fig. 3.14. It can be seen, only DGSAC-G and DGSAC-O are

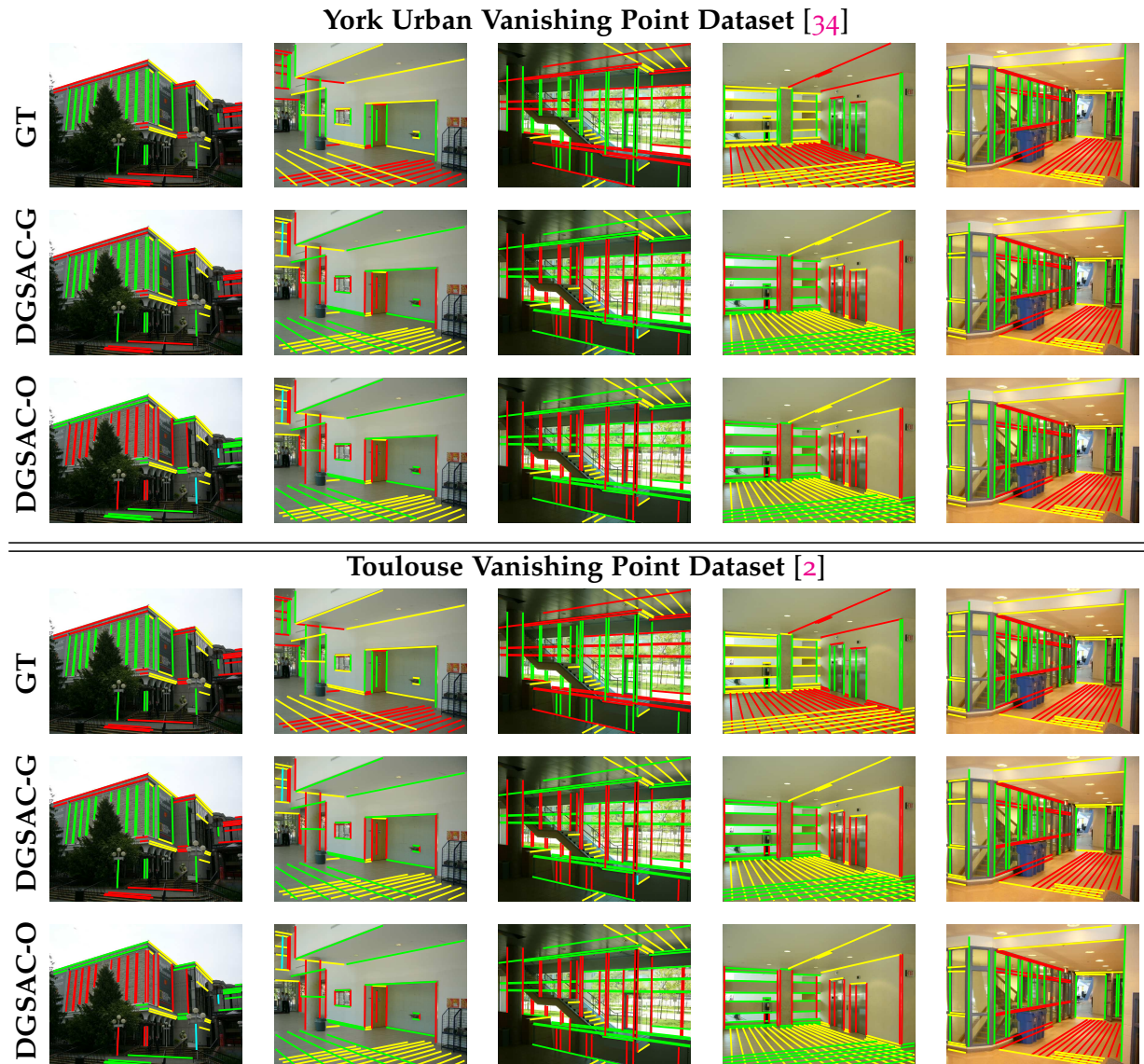


Figure 3.11: Sample Qualitative Results on York Urban and Toulouse Vanishing Point dataset.

able to recover all the five circles. The next best performing method is RansaCov, which is able to recover 4 out of 5 structures. J-Linkage leads to over-segmentation of structures, while T-Linkage is able to recover only two structures.

### 3.7 Discussion

The inlier noise scale and the ground-truth number of genuine structures present in the data are the two critical parameters of the multi-model fitting process. Most multi-model fitting methods require either or both of the above two parameters to be provided by the user, which introduce user dependency and limit the applicability where automatic execution is required. The best performing method NMU [137] is also susceptible to the user-provided inlier threshold (as presented in the original paper). In this work, we propose a data-driven unified pipeline for automatic robust multiple structure recovery.

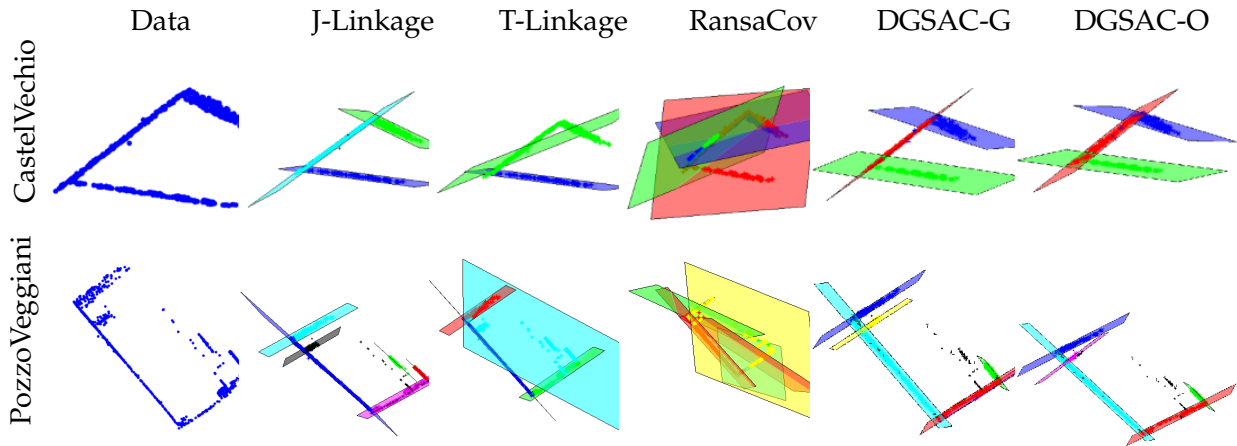


Figure 3.12: Multiple Plane Fitting to 3D Point Cloud. Dataset: CastelVecchio and PozzoVeggiani examples from SAMANTHA dataset [43]. Point membership is color coded.

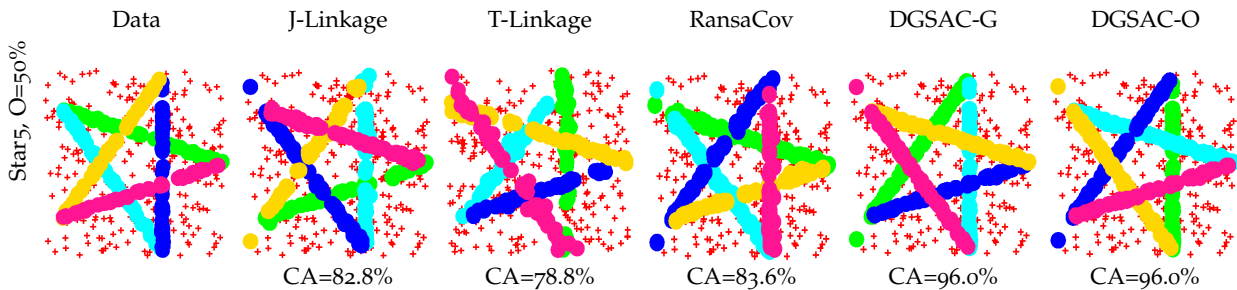


Figure 3.13: Multiple Line Fitting. Dataset: Star5 [144]. Point membership is color coded. Gross outliers are in red.

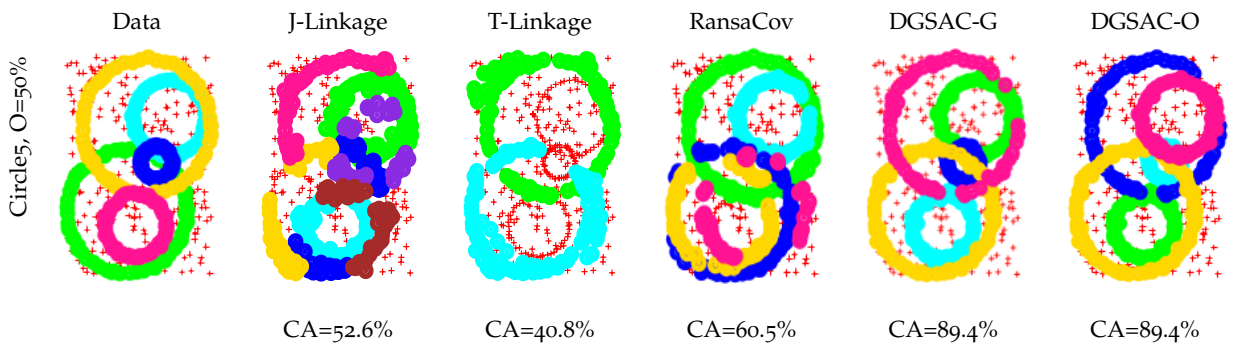


Figure 3.14: Multiple Circle Fitting. Dataset: Circle5 [144]. Point membership is color coded. Gross outliers are in red.

The proposed DGSAC utilizes kernel residual density to differentiate inliers and outliers. The KRD is applied to all components of the DGSAC pipeline. Using the KRD based guided sampling, DGSAC generates more relevant hypotheses and performs *greedy* or *optimization based* model selection by employing kernel density-based model hypotheses goodness measure. We believe that DGSAC plays a crucial role in the application that requires the automatic extraction of multiple structures. We plan further to improve the running time by parallel and optimized implementation.



## 4 Robust Image Classification

“The beginning of knowledge is the discovery of something we do not understand”

–Frank Herbert (1920 - 1986)

### 4.1 Introduction

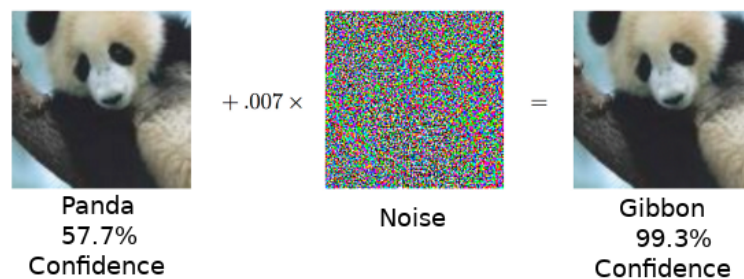


Figure 4.1: An adversarial example misclassified by an image Classifier

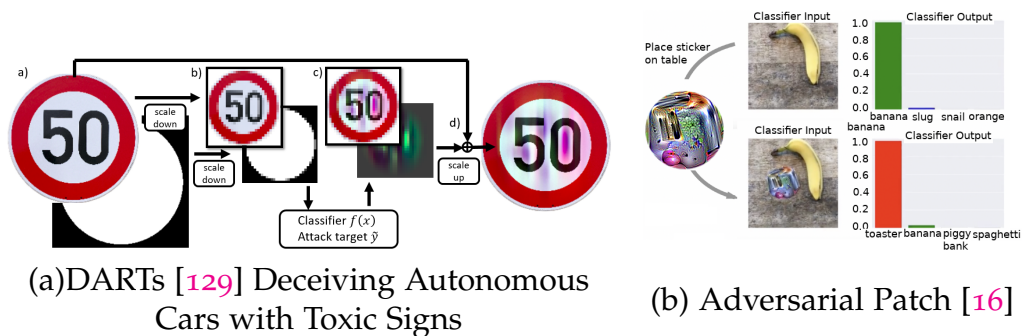


Figure 4.2: Some adversarial example generation systems.

Deep Neural Networks (DNN’s) have shown outstanding performance on many computer vision tasks such as image classification [73], speech recognition [64], and video classification [68]. Despite showing superhuman capabilities in the image classification task [63], the existence of *adversarial examples* [132] have raised questions on the reliability of neural network solutions for safety-critical applications.

An example of an adversarial perturbation is shown in Fig.4.1, where noise (middle) is added to the clean image (left most) to get an adversarial image (rightmost). Both the images visually look similar, however, the classifier misclassifies the perturbed, noisy image into “gibbon” with high confidence, while the un-perturbed image is classified into its true class “Panda”. The changes in adversarial examples are imperceptible to humans

but can lead to deep neural networks to make classification errors. These adversarial attacks are not restricted to the digital world, but can also exist in the physical world (Fig. 4.2).

Adversarial examples are carefully manipulated adaptations of an input, generated with the intent to fool a classifier into misclassifying them. Recently, it has been shown that adversarial examples are not limited to images but also exists in automatic speech recognition [122], text [9] and video [179] classification. In certain security-critical applications, failures of neural network-based models could lead to catastrophic outcomes. For example, an AI assistant incorrectly authenticating an impostor's voice and providing access to the confidential content [26], or a vision-based driver assistance system incorrectly recognizing a stop sign as a speed limit sign, [41], which may lead to fatal accidents.

One of the reasons for the attention that adversarial examples garnered is the ease with which they can be generated for a given model by simply maximizing the corresponding loss function. This is achieved by simply using a gradient based approach that finds a small perturbation at the input which leads to a large change in the output [132]. This apparent instability in neural networks is most pronounced for deep architectures that have an accumulation effect over the layers. This results in taking the small, additive, adversarial noise at the input and amplifying it to generating substantially noisy feature maps at intermediate layers that eventually influences the softmax probabilities enough to misclassify the perturbed input sample. This observation of amplification of input noise over the layers is not new, and has been pointed out in the past [132, 164]. The recent work by Xie et al. [164] addresses this issue by introducing *feature denoising* blocks in a network and training them with adversarial generated examples.

The iterative nature of generating adversarial examples makes their use in training to generate defenses computationally very expensive. For instance, the adversarially trained feature denoising model proposed by [164] takes 38 hours on 128 Nvidia V100 GPUs to train a baseline ResNet-101 with ImageNet. While we leverage this observation of noise amplification over the layers, our proposed approach *avoids any training or fine-tuning* of the model. Instead, we use a representative subset of training samples and their layer-wise pre-activation responses to construct nonparametric generative classifiers, which are then combined in an ensemble using ranking preferences.

Generative classifiers have achieved varying degrees of success as defense strategies against adversarial attacks [109, 82, 123]. Recently, [44] studied the class-conditional generative classifiers and concluded that it is impossible to guarantee robustness of such models. More importantly, they highlight the challenges in training generative classifiers using maximum likelihood based objective and their limitations w.r.t. discriminative ability and identification of out-of-distribution samples. While we propose to use generative classifiers, we avoid using likelihood based measures for making classification decisions. Instead, we use rank-order preferences of these classifiers which are then combined using a *Borda count*-based voting scheme. Borda counts have been used in collective decision making and are known to be robust to various manipulative attacks [120].

Most successful defense strategies adopt adversarial training or random input transformations that typically require retraining or fine-tuning the model to achieve reasonable performance. Our investigations of intermediate representations of a *pre-trained* DNN



lead to an interesting discovery pointing to intrinsic robustness to adversarial attacks. We find that we can learn a *generative* classifier by statistically characterizing the neural response of an intermediate layer to clean training samples. The predictions of multiple such intermediate-layer based classifiers, when aggregated, show unexpected robustness to adversarial attacks. Specifically, we devise an ensemble of these generative classifiers that rank-aggregates their predictions via a *Borda count*<sup>1</sup>-based consensus. Our proposed approach uses a subset of the clean training data and a pre-trained model, and yet is agnostic to network architectures or the adversarial attack generation method. We refer our defense against adversarial attacks on deep networks, referred to as *Rank-aggregating Ensemble of Generative classifiers for robust predictions* (REGroup). Consistent with recent trends, we focus only on the ImageNet dataset to evaluate the robustness of our defense and report performance superior to recent defenses that rely on adversarial training [75] and random input transformation [114] based approaches. Finally, we present extensive analysis of our defense with two different architectures (ResNet and VGG) on different targeted and untargeted attacks, restricted and unrestricted adversarial attacks. We show that our defense strategy achieves state-of-the-art performance on the ImageNet validation set.

### 4.1.1 Contributions

Our primary contributions are summarized below:

- We present REGroup, a retraining free, model-agnostic defense strategy that leverages an ensemble of generative classifiers over intermediate layers of the model.
- We model each generative classifier as a simple mixture distribution of neural responses obtained from a subset of training samples. We discover that *both positive and negative* pre-activation values contain information that can help correctly classify adversarially perturbed samples.
- We leverage the robustness inherent in Borda-count based consensus over the generative classifiers.
- We show extensive comparisons and analysis experiments on the ImageNet dataset spanning a variety of adversarial attacks.

## 4.2 Defense Approaches to Adversarial Attacks

Several defense techniques have been proposed to make neural networks robust to adversarial attacks. Broadly, we can categorize them into two approaches that: 1. Modify training procedure or modify input before testing; 2. Modify network architecture or add/change network hyper-parameters, optimization procedure, activation functions etc.

---

<sup>1</sup>Borda count is a voting mechanism where each voter rank all the candidates based on his preferences. The lowest-ranked candidate is given a low score for each ballot and a high score to the high ranked candidate. For each candidate, we aggregate scores across all ballots. The aggregated score is called the Borda count, and the candidate with the highest Borda count wins the election.

### 4.2.1 Modify Training and Modify Inputs During Testing

Some approaches of defenses in this category are mentioned below. *Adversarial training* [98, 184, 100] regularizes the neural network to reduce the over-fitting and in turn, improves the robustness. *Data compression* [10, 31] suppresses the high-frequency components and presents an ensemble-based defense approach. *Data randomization* [154, 163, 180] based approaches apply random transformations to the input to defend against adversarial examples by reducing their effectiveness.

However, it is shown in [100] and [128] that we can still generate adversarial examples for adversarially trained DNN's and for compression based defenses respectively.

### 4.2.2 Modify Network / Network Add-ons

Defenses under this category are either *detection only* or do both *detection and correction*. The aim of detection only defenses is to highlight if an example is adversarial and prevent it from further processing. These approaches include employing a detector sub-network [97], training the main classifier with an outlier class [57], using convolution filter statistics [80], or applying feature squeezing [167] to detect adversarial examples. However, all of these methods have shown to be ineffective against strong adversarial attacks [17, 124]. Full defense approaches include applying defensive distillation [107, 106] to use the knowledge from the output of the network to re-train the original model and improve the resilience of a network to small perturbations. Another approach is to augment the network with a sub-network called Perturbation Rectifying Network (PRN) to detect the perturbations; if the perturbation is detected, then PRN is used to classify the input image. However, later it was shown that Carlini and Wagner (C&W) attack [18] successfully defeated the defensive distillation approach.

### 4.2.3 Defenses for Large Scale Image Classification

ImageNet dataset [33] is considered one of the biggest dataset for image classification task. A few approaches have been proposed for the ImageNet dataset: Most of these approaches are based on input transformations or image denoising. Almost all the defenses designed for ImageNet have failed a thorough evaluation. A list of such defenses along with a thorough evaluation can be viewed at [90]. [113] and [83] claimed 81% and 75% ImageNet classification accuracy respectively under adversarial attacks. But after thorough evaluation [3] and accounting for obfuscated gradients [4], both accuracies were reduced to 0%. Similarly, [165] and [59] claimed 86% and 75% respectively, but these were also reduced to 0% [4]. A different approach proposed in [67] claimed accuracy 27.9% but later it was also reduced to 0.1% [39].

## 4.3 REGroup Methodology

Well-trained deep neural networks have a hierarchical structure, where the early layers transform inputs to feature spaces capturing local or more generic information, while later layers aggregate the local information to learn more semantically meaningful representations. In REGroup, we use many of the higher layers and learn class-conditional

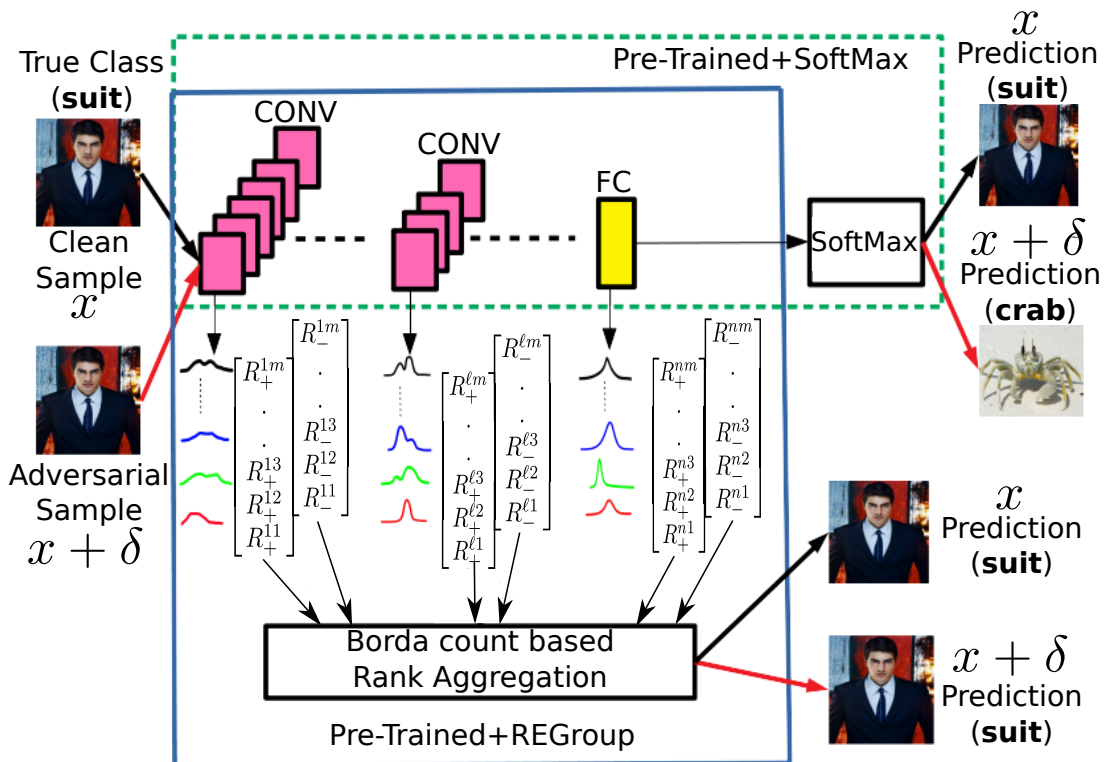


Figure 4.3: Overview of REGroup: Rank-aggregating Ensemble of Generative classifiers for robust predictions. REGroup uses a pre-trained network, and constructs layer-wise generative classifiers modeled by a mixture distribution of the positive and negative pre-activation neural responses at each layer. At test time, an input sample's ( $x$  or  $x + \delta$ ) neural responses are tested with generative classifiers to obtain ranking preferences ( $R_+$  and  $R_-$ ) of all  $m$  classes at all  $l$  layers. These preferences are aggregated across all layers using Borda count based preferential voting theory to make final prediction. Note: construction of layer-wise generative classifiers is a one time process.

generative classifiers as simple mixture-distributions estimated from the pre-activation neural responses at each layer from a subset of training samples. The generative classifier at each layer acts as an individual classifier. An ensemble of these layer-wise generative classifiers is used to make the final prediction by performing a Borda count-based rank-aggregation. Ranking preferences have been used extensively in robust fitting problems in computer vision [24, 23, 139], and we show its effectiveness in introducing robustness in DNNs against adversarial attacks.

Fig. 4.3 illustrates the overall working of REGroup. The approach has three main components: First, we use a layer as a generative classifier that produces a ranking predictions over all classes. The classifier assign high rank to a most probable class and lowest rank to least probable class. Second, each of these class-conditional generative classifiers are modeled using a mixture-distribution over the neural responses of the corresponding layer. Finally, the individual layer's class ranking preferences (ranked predictions) are aggregated using Borda count-based scoring to make the final predictions. We introduce the notation below and discuss each of these steps in detail in the subsections that follow.

**Notation:** In this chapter, we will *always* use  $\ell$ ,  $i$  and  $j$  for indexing the  $\ell^{\text{th}}$  layer,  $i^{\text{th}}$  feature

map and the  $j^{\text{th}}$  input sample respectively. The *true* and *predicted* class label will be denoted by  $y$  and  $\hat{y}$  respectively.

A classifier can be represented in a functional form as  $\hat{y} = \mathcal{F}(\mathbf{x})$ , it takes an input  $\mathbf{x}$  and predicts its class label  $\hat{y}$ . We define  $\phi^{\ell i}$  as the  $\ell^{\text{th}}$  layer's  $i^{\text{th}}$  pre-activation feature map, i.e., the neural responses *before* they pass through the activation function. For convolutional layers, this feature map  $\phi^{\ell i}$  is a 2D array, while for a fully connected layer, it is a scalar value.

## 4.4 DNN Layers as Generative Classifiers

We use the highest  $k$  layers of a DNN as generative classifiers that use the pre-activation neural responses to produce a ranking preferences over all classes. The layer-wise generative classifiers are modeled as a class-conditional mixture distribution, which is estimated using only a *pre-trained* network and a small subset  $\mathcal{S}$  of the training data. Let  $\mathcal{S}$  contain only correctly classified *training samples*<sup>2</sup>, which we can further divide into  $M$  subsets, one for each class i.e  $\mathcal{S} = \{\cup_{y=1}^M \mathcal{S}_y\}$ , where  $\mathcal{S}_y$  is the subset containing samples that have labels  $y$ .

### 4.4.1 Layerwise Neural Response Distributions

Our preliminary observations indicated that while the ReLU activations truncate the negative pre-activations during the forward pass, these values still contain semantically meaningful information. Our ablative studies in Fig. 4.9 confirm this observation and additionally, on occasion, we find that the negative pre-activations are complementary to the positive ones. Since the pre-activation features are real-valued, we compute the features  $\phi_j^{\ell i}$  for the  $j^{\text{th}}$  sample  $\mathbf{x}_j$ , and define its positive ( $P_j^{\ell i}$ ) and negative ( $N_j^{\ell i}$ ) response accumulators as  $P_j^{\ell i} = \sum \max(0, \phi_j^{\ell i})$ ,  $N_j^{\ell i} = \sum \max(0, -\phi_j^{\ell i})$ .

For convolutional layers, these accumulators represent the overall strength of positive and negative pre-activation responses respectively, when aggregated over the spatial dimensions of the  $i^{\text{th}}$  feature map of the  $\ell^{\text{th}}$  layer. On the other hand, for the linear layers, the accumulation becomes trivial with each neuron having a scalar response  $\phi_j^{\ell i}$ . We can now represent the  $\ell^{\text{th}}$  layer by the positive and negative response accumulator vectors denoted by  $P_j^\ell$  and  $N_j^\ell$  respectively. We normalize these vectors and define the layer-wise probability mass function (PMF) for the positive and negative responses as  $\mathbb{P}_j^\ell = \frac{P_j^\ell}{\|P_j^\ell\|_1}$

and  $\mathbb{N}_j^\ell = \frac{N_j^\ell}{\|N_j^\ell\|_1}$  respectively.

Our interpretation of  $\mathbb{P}_j^\ell$  and  $\mathbb{N}_j^\ell$  as a PMF could be justified by drawing an analogy to the softmax output, which is also interpreted as a PMF. However, it is worth emphasizing that we chose the linear rescaling of the accumulator vectors rather than directly applying a softmax normalization. By separating out the positive and negative accumulators, we obtain two independent representations for each layer, which is beneficial to our

<sup>2</sup>We took 50,000 out of  $\sim 1.2$  millions training images from ImageNet dataset, 50 per class.

rank-aggregating ensemble discussed in the following sections. A softmax normalization over a feature map comprising of positive and negative responses would have entirely suppressed the negative responses, discarding all its constituent semantic information. An additional benefit of the linear scaling is its simple computation. Algorithm 6 summarizes the computation of the layer-wise PMFs for a given training sample.

---

**Algorithm 6:** Layerwise PMF of neural responses.  $H \times W$  represents the spatial dimensions of pre-activation features. For  $\ell^{\text{th}}$  convolutional layer the dimensions of feature maps  $H \times W = r^\ell \times s^\ell$ , and for linear layers the dimensions of neuron output  $H \times W = 1 \times 1$ .

---

```

1 Input:  $\mathbf{x}_j$  pre-activation features  $\boldsymbol{\phi}_j^{\ell i} \in \mathbb{R}^{H \times W}$ 
2 for  $\ell \in [1..n]$  do
3    $P_j^{\ell i} = \sum \max(0, \boldsymbol{\phi}_j^{\ell i}), \quad \forall i$  (sum over H, W)
4    $N_j^{\ell i} = \sum \max(0, -\boldsymbol{\phi}_j^{\ell i}), \quad \forall i$  (sum over H, W)
5 end
6  $P_j^\ell \leftarrow P_j^\ell + \delta, \quad N_j^\ell \leftarrow N_j^\ell + \delta$ 
7  $\mathbb{P}_j^{\ell i} \leftarrow \frac{P_j^{\ell i}}{\sum_i P_j^{\ell i}}, \quad \mathbb{N}_j^{\ell i} \leftarrow \frac{N_j^{\ell i}}{\sum_i N_j^{\ell i}}$  (PMFs)

```

---

#### 4.4.2 Layerwise Generative Classifiers

We model the layerwise generative classifiers for class  $y$  as a class-conditional mixture of distributions, with each mixture component as the PMFs  $\mathbb{P}_j^\ell$  and  $\mathbb{N}_j^\ell$  for a given training sample  $\mathbf{x}_j \in \mathcal{S}_y$ . The generative classifiers corresponding to the positive and negative neural responses are then defined as the following mixture of PMFs

$$\mathbf{C}_y^{+\ell} = \sum_{j:\mathbf{x}_j \in \mathcal{S}_y} \lambda_j \mathbb{P}_j^\ell, \quad \mathbf{C}_y^{-\ell} = \sum_{j:\mathbf{x}_j \in \mathcal{S}_y} \lambda_j \mathbb{N}_j^\ell \quad (4.1)$$

where the weights  $\lambda_j$  are nonnegative and add up to one in the respective equations. We choose the weights to be proportional to the softmax probability value as predicted by the network given the input  $\mathbf{x}_j$ . Using the subset of training samples  $\mathcal{S}$ , we construct the class-conditional mixture distributions,  $\mathbf{C}_y^{+\ell}$  and  $\mathbf{C}_y^{-\ell}$  at each layer  $\ell$  only once. At inference time, we input a test sample  $\mathbf{x}_j$ , from the test set  $\mathcal{T}$ , to the network and compute the PMFs  $\mathbb{P}_j^\ell$  and  $\mathbb{N}_j^\ell$  using Algorithm 6. As our test input is a PMF and the generative classifier is also a mixture distribution, we simply use the KL-Divergence between the classifier model  $\mathbf{C}^{+\ell}$  and the test sample  $\mathbb{P}_j^\ell$  as a classification score as

$$P_{KL}(\ell, y) = \sum_i \mathbf{C}_y^{+\ell i} \log \left( \frac{\mathbf{C}_y^{+\ell i}}{\mathbb{P}_j^{\ell i}} \right), \forall y \in \{1, \dots, M\} \quad (4.2)$$

and similarly for the negative PMFs

$$N_{KL}(\ell, y) = \sum_i C_y^{-\ell i} \log \left( \frac{C_y^{-\ell i}}{\mathbb{N}^{\ell i}} \right), \forall y \in \{1, \dots, M\} \quad (4.3)$$

We use a simple classification rule and select the predicted class  $\hat{y}$  as the one with the smallest KL-Divergence with the test sample PMF. However, rather than identifying  $\hat{y}$ , at this stage we are only interested in rank-ordering the classes, which we simply achieve by sorting the KL-Divergences (Eqns. (4.2) and (4.3)) in ascending order. The resulting ranking of classes for the  $\ell^{\text{th}}$  layer are given below in Eqns. (4.4) and (4.5) respectively. Where,  $R_+^{\ell y}$  is the rank (position of  $y^{\text{th}}$  class in the ascending order of KL-Divergences in  $P_{KL}$ ) of  $y^{\text{th}}$  class in the  $\ell^{\text{th}}$  layer preference list  $R_+^{\ell}$ .

$$R_+^{\ell} = [R_+^{\ell 1}, R_+^{\ell 2}, \dots, R_+^{\ell y}, \dots, R_+^{\ell M}] \quad (4.4)$$

$$R_-^{\ell} = [R_-^{\ell 1}, R_-^{\ell 2}, \dots, R_-^{\ell y}, \dots, R_-^{\ell M}] \quad (4.5)$$

## 4.5 Robust Predictions with Rank Aggregation

Rank aggregation based preferential voting for making group decisions is widely used in selecting a winner in a democratic setup [120]. The basic premise of preferential voting is that  $n$  voters are allowed to rank  $m$  candidates in the order of their preferences. The rankings of all  $n$  voters are then aggregated to make a final prediction.

Borda count [13] is one of the approaches for preferential voting that relies on aggregating the rankings of all the voters to make a collective decision [120, 66]. The other popular voting strategies to find a winner out of  $m$  different choices include Plurality voting [148], and Condorcet winner [177]. In Plurality voting, the winner would be the one who gets the maximum fraction of votes, while Condorcet winner is the one who gets the majority votes.

### 4.5.1 Rank Aggregation using Borda Count

*Borda count* is a generalization of the majority voting. In a two-candidates case it is equivalent to majority vote. The *Borda count* for a candidate is the sum of the number of candidates ranked below it by each voter. In our setting, while processing a test sample  $\mathbf{x}_j \in \mathcal{T}$ , every layer acts as two independent voters based on  $\mathbb{P}^{\ell}$  and  $\mathbb{N}^{\ell}$ . The number of classes i.e  $M$  is the number of candidates. The Borda count for the  $y^{\text{th}}$  class at the  $\ell^{\text{th}}$  layer is denoted by  $B^{\ell y} = B_+^{\ell y} + B_-^{\ell y}$ , where  $B_+^{\ell y}$  and  $B_-^{\ell y}$  are the individual Borda count of both the voters and computed as shown in equation (4.6).

$$B_+^{\ell y} = (M - R_+^{\ell y}), \quad B_-^{\ell y} = (M - R_-^{\ell y}) \quad (4.6)$$

### 4.5.2 Hyperparameter Settings

We aggregate the Borda counts of highest  $k$  layers of the network, which is the only hyperparameter to set in REGroup. Let  $B^{:ky}$  denote the aggregated Borda count of  $y^{th}$  class from the last  $k$  layers irrespective of the type (convolutional or fully connected). Here,  $n$  is the total number of layers. The final prediction would be the class with maximum aggregated Borda count.

$$\begin{aligned}
 B^{:ky} &= \sum_{\ell=n-k+1}^n B^{\ell y} \\
 &= \sum_{\ell=n-k+1}^n B_+^{\ell y} + B_-^{\ell y}, \quad \forall y \in \{1..M\} \\
 \hat{y} &= \operatorname{argmax}_y B^{:ky}
 \end{aligned} \tag{4.7}$$

To determine the value of  $k$ , we evaluate REGroup on 10,000 *correctly classified* samples from the ImageNet Validation set at each layer, using per layer Borda count i.e  $\hat{y} = \operatorname{argmax}_y B^{\ell y}$ . We select  $k$  to be the number of later layers at which we get at-least 75% accuracy. This can be viewed in the context of the confidence of individual layers on discriminating samples of different classes. We follow the above heuristic and found  $k = 5$  for both the architectures ResNet-50 and VGG-19, which we use in all our experiments. An ablation study with all possible values of  $k$  is included in section 4.6.6.

## 4.6 Experimental Analysis

In this section, we evaluate robustness of REGroup against state-of-the-art attack methods. We follow the recommendations on defense evaluation in [19].

### 4.6.1 Adversarial Attacks

We consider attack methods in the following two categories: *gradient-based* and *gradient-free*.

#### 4.6.1.1 Gradient-Based Attacks

In this category, we consider two variants, *restricted* and *unrestricted* attacks. The restricted attacks generate adversarial examples by searching an adversarial perturbations within the bound of  $L_p$  norm, while unrestricted attacks generate adversarial example by manipulating image-based visual descriptors. Due to restriction on the perturbation the adversarial examples generated by restricted attacks are similar to the clean original image, while unrestricted attacks generate natural-looking adversarial examples, which are far from the clean original image in terms of  $L_p$  distance. We consider the following, *Restricted attacks*: Projected Gradient Descent (PGD) [91], DeepFool [99], Carlini and Wagner (C&W) [18] and Trust Region [174], and *Unrestricted attack*: cAdv [11] semantic manipulation attack. The attacks in the restricted category generate adversarial examples by searching an adversarial perturbations within the bound of  $L_p$  norm. Due to this

restriction on perturbation the generated examples are close to the original image. In the unrestricted attacks category the adversarial examples are generated by semantically manipulating image-based visual descriptors. The semantic manipulation attack in cAdv[11] generates natural-looking adversarial examples, which are far from the original image in the terms of  $L_p$  distance (hence a large unrestricted perturbation). An example of cAdv is shown in Fig. 4.4.



Figure 4.4: cAdv [11] adversarial examples

#### 4.6.1.2 Gradient-Free Attacks

The approaches in this category do not have access to the network weights. We consider following attacks: SPSA [147], Boundary [15] and Spatial [40].

### 4.6.2 Experimental Setup

**Architectures:** We use two different network architectures ResNet-50<sup>3</sup> and VGG-19<sup>4</sup>, both with ImageNet *pre-trained* weights.

**Datasets:** We present our evaluations, comparisons and analysis only on ImageNet [33] dataset. We use the subsets of full ImageNet validation set as described in Tab. 4.1. Note that, V10K, V2K and V10C would be different for ResNet-50 and VGG-19, since an image classified correctly by ResNet-50 need not be classified correctly by the VGG-19.

Dataset	Description
V50K	Full ImageNet validation set with 50000 images.
V10K	A subset of 10000 correctly classified images from V50K set. 10 Per class.
V2K	A subset of 2000 correctly classified images from V50K set. 2 Per class.
V10C	A subset of correctly classified images of 10 sufficiently different classes.

Table 4.1: Dataset used for evaluation and analysis.

<sup>3</sup><https://download.pytorch.org/models/resnet50-19c8e357.pth>

<sup>4</sup><https://download.pytorch.org/models/vgg19-dcbb9e9d.pth>



### 4.6.3 Performance on Gradient-Based Attacks

#### 4.6.3.1 Comparison with adversarial-training / fine-tuning

We evaluate REGroup on clean samples as well as adversarial examples generated using PGD ( $\epsilon = 16$ ) from V50K dataset, and compare it with prior state-of-the-art works. The results are reported in Tab. 4.2, and we see that REGroup outperforms the state-of-the-art input transformation based defense BaRT [114], both in terms of the clean and adversarial samples (except in the case of Top-1 accuracy with  $k = 10$ , which is the number of input transformations used in BaRT). We see that while our performance on clean samples decreases when compared to adversarial training (Inception v3), it improves significantly on adversarial examples with a high  $\epsilon = 16$ . While our method is not directly comparable with adversarially trained Inception v3 and ResNet-152, because the base models are different, a similar decrease in the accuracy over clean samples is reported in their paper. The trade-off between robustness and the standard accuracy has been studied in [35] and [146].

An important observation to make with this experiment is, if we set aside the base models of ResNets and compare Top-1 accuracies on clean samples of full ImageNet validation set, our method (REGroup) without any *adv-training/fine-tuning* either outperforms or performs similar to the state-of-the-art *adv-training/fine-tuning* based methods [114, 164].

(Dataset used: ImageNet-V50K). Model	Clean Images		Attacked Images	
	Top-1	Top-5	Top-1	Top-5
ResNet-50	76	93	0.0	0.0
Inception v3	78	94	0.7	4.4
ResNet-152	79	94	-	-
Inception v3 w/ Adv. Train	78	94	1.5	5.5
ResNet-152 w/ Adv. Train	63	-	45	-
ResNet-152 w/ Adv. Train w/ denoise	66	-	49	-
ResNet-50-BaRT, $k = 5$	65	85	16	51
ResNet-50-BaRT, $k = 10$	65	85	36	57
ResNet-50-REGroup	66	86	22	65

Table 4.2: Comparison with adversarially trained and fine-tuned classification models. Top-1 and Top-5 classification accuracy (%) of adversarial trained (Inception V3 [75] and ResNet-152 [164]) and fine-tuned (ResNet-50 BaRT [114]) classification models. Clean Images are the non-attacked original images. The results are divided into three blocks, the top block include original networks, middle block include defense approaches based on adversarial re-training/fine-tuning of original networks, bottom block is our defense without re-training/fine-tuning. Results of the competing methods are taken from their respective papers. '-' indicate the results were not provided in the respective papers.

#### 4.6.3.2 Performance w.r.t PGD adversarial strength

We evaluate REGroup w.r.t the maximum perturbation of the adversary. The results are reported in Fig. 4.5(a). REGroup outperforms both the adversarial training [75] and

BaRT [114]. Both adversarial training and BaRT have shown protection against PGD adversarial attacks with a maximum perturbation strength  $\epsilon = 16$  and  $\epsilon = 32$  respectively, however we additionally show the results with  $\epsilon = 40$  on full ImageNet validation set. We also note that with increasing perturbation strength, our defense’s accuracy is also strictly decreasing. This is in accordance with [19], where transitioning from a clean image to noise should yield a downward slope in accuracy, else there could be some form of gradient masking involved. While it may seem  $\epsilon = 40$  is a large perturbation budget and it will destroy the object information in the image completely, but we would like to emphasize that it is not the case when using large size images. A comparison of PGD examples generated with  $\epsilon = 40$  using CIFAR-10 ( $32 \times 32$ ) and ImageNet ( $224 \times 224$ ) images is shown in Fig. 4.5(b).

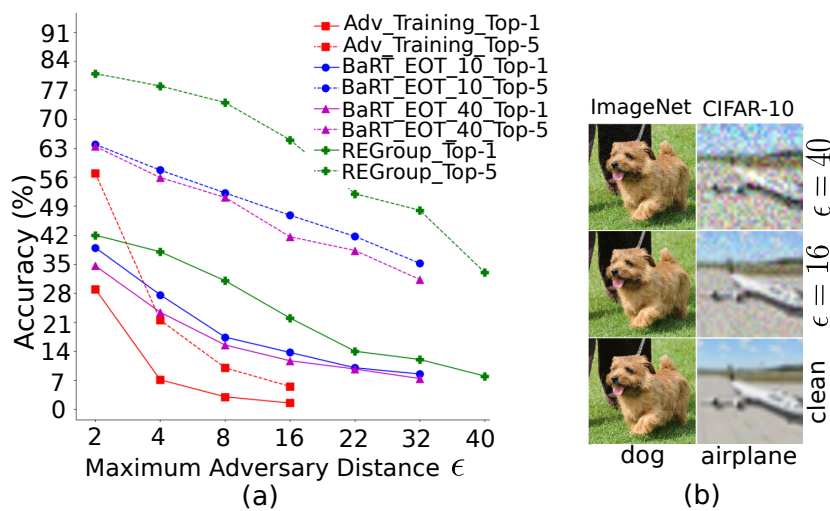


Figure 4.5: Top-1 and Top-5 accuracy(%) w.r.t PGD adversarial strength. Comparison with adversarial training based method [75] and fine-tuning using random input transformations based method (BaRT) [114] with Expectation Over Transformation (EOT) steps 10 and 40, against the PGD perturbation strength ( $\epsilon$ ). The results of the competing methods are taken from their respective papers. Dataset used: ImageNet-V50K.

#### 4.6.3.3 Performance on un-targeted attacks.

We evaluate REGroup on various untargeted attacks and report results in Tab. 4.3. The perturbation budgets ( $\epsilon$ ) and dataset used for the respective attacks are listed in the table. With the exception of the maximum perturbation allowed, we used default parameters given by FoolBox [117]. We observe that the performance of our defense is quite similar for both the models employed. This is due to the attack-agnostic nature of our defense. We achieve 48% accuracy (ResNet-50) for PGD attack using our defense which is significant given that PGD is considered to be one of the strongest attacks among the class of first order adversaries.

	Data	UN /		ResNet-50			VGG-19		
		TA / HC	$\epsilon$	SMax		REGroup	SMax		REGroup
				#S	T1(%)	T1(%)	#S	T1(%)	T1(%)
Clean	V10K	-	-	10000	100	88	10000	100	76
Clean	V2K	-	-	2000	100	86	2000	100	72
Clean	V10C	-	-	417	100	84	392	100	79
PGD	V10K	UN	4 ( $L_\infty$ )	9997	0	48	9887	0	46
DFool	V10K	UN	2 ( $L_2$ )	9789	0	61	9939	0	55
C&W	V10K	UN	4 ( $L_2$ )	10000	0	40	10000	0	38
TR	V10K	UN	2 ( $L_\infty$ )	10000	0	41	9103	0	45
cAdv	V10C	UN	-	417	0	37	392	0	18
PGD	V2K	TA	( $L_\infty$ )	2000	0	47	2000	0	31
C&W	V2K	TA	( $L_2$ )	2000	0	46	2000	0	38
PGD	V2K	UN+HC	( $L_\infty$ )	2000	0	21	2000	0	19
PGD	V2K	TA+HC	( $L_\infty$ )	2000	0	23	2000	0	17

Table 4.3: Performance on Gradient-Based Attacks. Comparison of Top-1 classification accuracy between SoftMax (SMax) and REGroup based final classification. UN and TA indicates, un-targeted and targeted attacks respectively. The +HC indicates adversarial examples are generated with high-confidence ( $> 90\%$ ) constraint, in this case  $\epsilon$  can be any value that satisfies the HC criteria. For targeted attack we select a target class uniformly at random from the 1000 classes leaving out the true class. #S is the number of images for which the attacker is successfully able to generate adversarial examples using the respective attack models and the accuracies are reported with respect to the #S samples, hence the 0% accuracies with the SoftMax (SMax). Since #S is different for several attacks, therefore, the performance may not be directly comparable across different attacks. ‘-’ indicate the information is not-applicable. For Data description refer Tab. 4.1.

#### 4.6.4 Performance on unrestricted, untargeted semantic manipulation attack.

We consider V10C dataset for cAdv attack. We use the publicly released source code by the authors.



Figure 4.6: **Unrestricted Adversarial Example Via Semantic Manipulation [11]. Row-1 Original Examples, Row-2 Semantically Perturbed Adversarial Examples.**

In this section, we evaluate REGroup against semantically perturbed unrestricted adversarial examples. Restricted adversarial examples are the examples which are carefully crafted by adding a small magnitude of perturbations. These perturbations are restricted within the  $\ell_\infty$  norm. Most of the defenses make use of this information to devise defensive strategy. A new method (called *cAdv*) for generating unrestricted adversarial examples is proposed in [11], that manipulate semantically meaningful image-based visual descriptors *e.g.* color. Semantically manipulated images affects the image classification and captioning tasks. A sample of such adversarial examples is shown in Fig. 4.6. Specifically we use *cAdv*<sub>4</sub> variant with the parameters suggested by the authors. The results are reported in Tab. 4.3.

#### 4.6.4.1 Performance on targeted attacks

We consider V2K dataset for targeted attacks and report the performance on PGD and C&W targeted attacks in Tab. 4.3.

#### 4.6.4.2 Performance on PGD attack with High Confidence

We evaluate REGroup on PGD examples on which the network makes highly confident predictions using SoftMax. We generate un-targeted and targeted adversarial examples using PGD attack with a constraint that the network’s confidence of the prediction of adversarial examples is at-least 90%. For this experiment we do not put constraint on the adversarial perturbation *i.e.*  $\epsilon$ . Results are reported in Tab. 4.3.

#### 4.6.4.3 Performance on Physically Realizable Attacks

In [162] a physically realizable attack method called rectangular occlusion attack (ROA) is proposed. In this attack method, an adversary place a small rectangle anywhere in the image and add the  $\ell_\infty$  noise within that rectangle. A gradient based search/Exhaustive search is adopted to select the best location to place the rectangle in the image. An example of physical attack example is shown in Fig. 4.7. The ROA attack is a attack method that realizes the physical attack in the digital space.

We apply ROA attack on V10K dataset and evaluate REGroup on the adversarial images. The results are shown in Tab. 4.4.

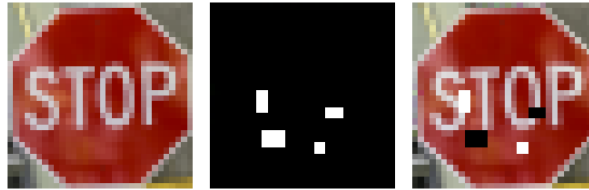


Figure 4.7: *Physical Attack [162]. Stop sign image with adversarial stickers, classified as a speed limit sign. Left: Original stop sign, Middle: Adversarial Mask, Right: Stop Sign classified as speed limit sign.*

Attack	#S	ResNet-50	
		SMax T1(%)	REGroup T1(%)
Physical Attack [162]	7939	0	17.3

Table 4.4: *Performance on Physical Adversarial Examples. Dataset: V10K. Top-1 ( %) classification accuracy comparison between SoftMax (SMax) and REGroup. #S is the number of images for which the attacker is successfully able to generate adversarial examples using the respective attack models and the accuracies are reported with respect to the #S samples, hence the 0% accuracies with the SoftMax (SMax).*

#### 4.6.5 Performance on Gradient-Free Attacks

Several studies [4], [108] have observed a phenomenon called *gradient masking*. This phenomenon occurs when a practitioner unintentionally or intentionally proposes a defense which does not have meaningful gradients, either by reducing them to small values (vanishing gradients), removing them completely (shattered gradients) or adding some noise to it (stochastic gradient). Gradient masking based defenses hinder the gradient computation and in turn inhibit gradient-based attacks, thus providing a false sense of security. Therefore, to establish the robustness of a defense against adversarial attacks in general, it is important to rule out that a defense relies on gradient masking.

To ensure that REGroup is not masking the gradients we follow the standard practice [105] [182] and evaluate on strong gradient-free SPSA [147] attack. In addition to SPSA, we also show results on two more gradient-free attacks, Boundary [15] and Spatial [40] attack. The results are reported in Tab. 4.5.

The consistent superior performance on both gradient-based (both restricted and unrestricted) and gradient free attack shows REGroup is not masking the gradients and is attack method agnostic.

	Data	UN /		ResNet-50			VGG-19		
		TA	HC	#S	SMax REGroup		#S	SMax REGroup	
		$\epsilon$	T1(%)		T1(%)	T1(%)		T1(%)	
SPSA	V10K	UN	4 ( $L_\infty$ )	4911	0	71	5789	0	58
Boundary	V10K	UN	2 ( $L_2$ )	10000	0	50	10000	0	50
Spatial	V10K	UN	2 ( $L_2$ )	2624	0	36	2634	0	30

Table 4.5: Performance on Gradient-Free Attacks. Top-1 (%) classification accuracy comparison between SoftMax (SMax) and REGroup. Legends are same as in Tab. 4.3.

## 4.6.6 Analysis and Ablation Study

### 4.6.6.1 Accuracy vs number of layers ( $k$ )

We report performance of REGroup on various attacks reported in Tab. 4.3 for all possible values of  $k$ . The accuracy of VGG-19 w.r.t. the various values of  $k$  is plotted in Fig. 4.8. We observe a similar accuracy vs  $k$  graph for ResNet-50 and note that a reasonable choice of  $k$  made based on this graph does not significantly impact REGroup’s performance. Refer Fig. 4.8, the ‘Agg’ stands for using aggregated Borda count  $B^{:ky}$ . PGD(V10K,UN), DFool, C&W(V10K,UN) and Trust Region are the same experiments as reported in Tab. 4.3, but with all possible values of  $k$ . ‘Per\_Layer\_V10K’ stands for evaluation using per layer Borda count i.e  $\hat{y} = \operatorname{argmax}_y B^{\ell y}$  on a separate 10,000 correctly classified subset of validation set. In all our experiments we choose the  $k$ -highest layers where ‘Per\_Layer\_V10K’ has at-least 75% accuracy. A reasonable change in this accuracy criteria of 75% would not affect the results on adversarial attacks significantly. However, a substantial change (to say 50%) deteriorates the performance on clean sample significantly. The phenomenon of decrease in accuracy of clean samples vs robustness has been studied in [35] and [146].

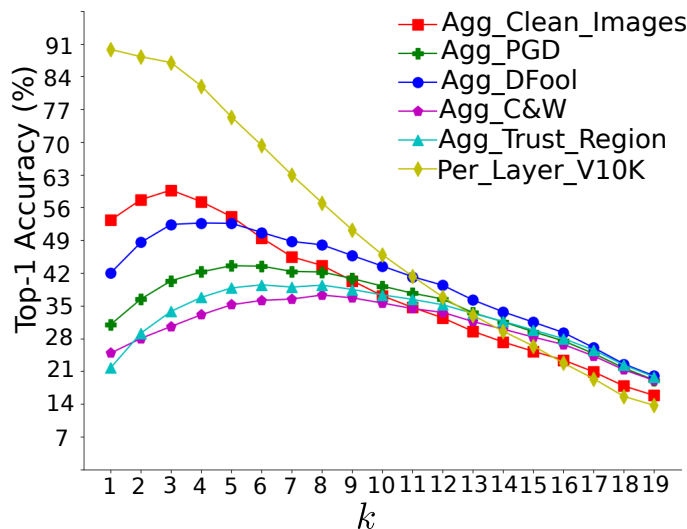


Figure 4.8: Ablation study. Accuracy vs no. of layers ( $k$ ).

#### 4.6.6.2 Effect of positive and negative pre-activation responses.

We report the impact of using positive, negative and a combination of both pre-activation responses on the performance of REGroup in Fig. 4.9. We consider three variants of Borda count rank aggregation from later  $k$  layers. Pos:  $B^{:ky} = \sum_{\ell=n-k+1}^n B_+^{\ell y}$ , Neg:  $B^{:ky} = \sum_{\ell=n-k+1}^n B_-^{\ell y}$ , and Pos+Neg:  $B^{:ky} = \sum_{\ell=n-k+1}^n B_+^{\ell y} + B_-^{\ell y}$ . We report the Top-1 accuracy (%) of the attacks experiment as set up in Tab. 4.3 (DF: DFool, C&W, TR: Trust Region), in Tab. 4.5 (BD: Boundary, SP: Spatial), and in Fig. 4.5 (PGD<sub>2</sub>, PGD<sub>4</sub> and PGD<sub>8</sub>, with  $\epsilon = 2, 4$  and  $8$  respectively). From the bar chart it is evident that in some experiments, Pos performs better than Neg (e.g UN\_TR), while in others Neg is better than Pos only (e.g UN\_DF). It is also evident that Pos+Neg occasionally improve the overall performance, and the improvement seems significant in the targeted C&W attacks for both the ResNet-50 and VGG-19. We leave it to the design choice of the application, if inference time is an important parameter, then one may choose either Pos or Neg to reduce the inference time to approximately half of what is reported in Tab. 4.6.

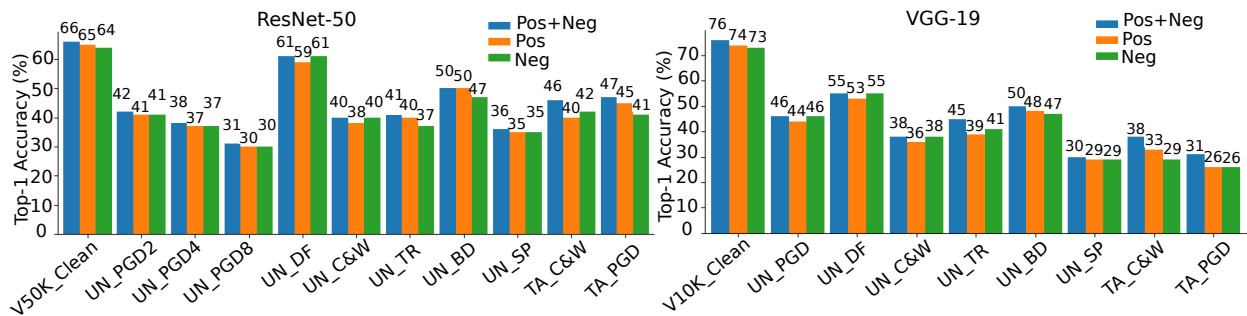


Figure 4.9: Ablation study. Effect of considering positive and negative pre-activation responses.

#### 4.6.6.3 Inference time using REGroup

: We use PyTorch for all our experiments and a GPU is only required for extracting layer outputs and adversarial example generation. Since we suggest to use REGroup during test time, we compare the inference time with SoftMax for both ResNet-50 and VGG-19 experiments on both GPU and CPU. The inference time is reported in table 4.6.

	ResNet-50				VGG-19			
	SMax		REGroup		SMax		REGroup	
	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU
Time(s)	0.02	0.06	0.13	0.35	0.03	0.12	0.16	0.64

Table 4.6: Inference Time Comparison. REGroup vs SoftMax: We use a workstation with an i7-8700 CPU and GTX 1080 GPU.

## 4.7 Discussion

In this work, we have presented a *simple*, *scalable*, and *practical* defense strategy that is model agnostic and does not require any *re-training* or *fine-tuning*. We suggested to use REGroup at test time to make a pre-trained network robust to adversarial perturbations.

Using challenging adversarial attacks created on ImageNet, we showed that the proposed defense, REGroup, performed competitively in comparison to state-of-the-art defenses [114, 75] that have a clear advantage of adversarial training / fine-tuning the base network. There are three main reasons that justify the success of REGroup. Firstly, instead of using a maximum likelihood based prediction, REGroup adopts a *ranking preference* based approach. Secondly, aggregation of preferences from multiple layers leads to group decision making, unlike SoftMax that relies on the output of the last layer only. Thirdly, there exists inherent robustness of Borda count in rank aggregation. It is well established that Borda count is robust to noise in rankings of individual voters [120], [66]. Hence, where SoftMax fails to predict the correct class of an adversarial example image generated by an attacker with an aim to misclassify in a maximum-likelihood sense, REGroup takes ranked predictions from multiple layer-wise generative classifiers and builds a consensus using Borda count to make final robust prediction. Our promising empirical results indicate that deeper theoretical analysis of REGroup would be an interesting direction to pursue.



## 5 Conclusion and Future Directions

In this dissertation, we have proposed robust solutions to three important scene understanding tasks. All three solutions are data-driven and hence require zero or negligible human intervention.

A self-supervised, self-improving geometric-CNN framework is proposed in chapter 2 for robust 3D perception. We demonstrated that the coupling of monocular geometric SLAM and unsupervised monocular depth prediction networks helps mitigate the shortcomings of each other by leveraging each other's strengths. We proposed a joint narrow and wide baseline based learning system that improves depth estimate, especially of the farther away points. Our concept of running Pseudo RGB-D SLAM provides a more robust and accurate solution as compared to RGB-SLAM and reduces the chances of tracking failure. A future extension of this work could be to design an online, real-time version of the current framework and extend it to more challenging settings *e.g.* uncalibrated and rolling shutter cameras [187, 188].

In chapter 3, we proposed a data-driven robust multiple model fitting framework for the simultaneous estimation of multiple geometric models. We leveraged Kernel Residual Density as a primary tool to design an automatic guided sampling algorithm with self-terminating criteria, to estimate the inlier noise scale, and to design a quadratic optimization program for model selection. The DGSAC pipeline plays an important role in the application that requires the automatic extraction of multiple structures. An important extension of this work would be to explore the deep learning directions to solve the multiple model fitting problems [72, 14, 183, 115].

In chapter 4, we proposed a test time replacement of SoftMax to convert a pre-trained classifier into a robust classifier. We introduced a *simple, scalable, and practical* approach towards making the image classification task robust to several adversarial attacks without any re-training or adversarial training. Extending the current approach to other modalities *e.g.* text and audio, and more complicated neural network architectures *e.g.* RNN would be another avenue for future works.

The integration of the proposed solutions into an intelligent visual perception system would enhance the system's robustness as a whole and help make the more complicated decisions more robustly and accurately. We believe this thesis achieves its aim of making the scene understanding tasks more robust to the noisy and corrupted real-world visual data.



# Bibliography

- [1] Lorenzo Andraghetti et al. “Enhancing Self-Supervised Monocular Depth Estimation with Traditional Visual Odometry”. In: *International Conference on 3D Vision (3DV)*. 2019, pp. 424–433.
- [2] Vincent Angladon, Simone Gasparini, and Vincent Charvillat. “The Toulouse Vanishing Points Dataset”. In: *ACM Multimedia Systems Conference (MMSys '15)*. 2015, pp. 231–236.
- [3] Anish Athalye and Nicholas Carlini. “On the Robustness of the CVPR 2018 White-Box Adversarial Example Defenses”. In: *arXiv preprint arXiv:1804.03286* (2018).
- [4] Anish Athalye, Nicholas Carlini, and David Wagner. “Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples”. In: *International Conference on Machine Learning (ICML)*. 2018, pp. 274–283.
- [5] Anish Athalye et al. “Synthesizing Robust Adversarial Examples”. In: *International Conference on Machine Learning (ICML)*. 2018, pp. 284–293.
- [6] Daniel Barath and Jiri Matas. “Progressive-X: Efficient, Anytime, Multi-Model Fitting Algorithm”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2019, pp. 3780–3788.
- [7] Daniel Barath, Jiri Matas, and Jana Noskova. “MAGSAC: Marginalizing Sample Consensus”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 10197–10205.
- [8] Harry Barrow et al. “Recovering intrinsic scene characteristics”. In: *Comput. Vis. Syst* 2.3-26 (1978), p. 2.
- [9] Melika Behjati et al. “Universal Adversarial Attacks on Text Classifiers”. In: *IEEE ICASSP*. 2019, pp. 7345–7349.
- [10] Arjun Nitin Bhagoji et al. “Enhancing Robustness of Machine Learning Systems via Data Transformations”. In: *Annual Conference on Information Sciences and Systems (CISS)*. 2018, pp. 1–5.
- [11] Anand Bhattad et al. “Unrestricted Adversarial Examples via Semantic Manipulation”. In: *International Conference on Learning Representations (ICLR)*. 2019.
- [12] Jia-Wang Bian et al. “Unsupervised Scale-Consistent Depth and Ego-Motion Learning from Monocular Video”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).
- [13] Duncan Black et al. “The Theory of Committees and Elections”. In: Cambridge University Press., 1958.
- [14] Eric Brachmann and Carsten Rother. “Neural-guided RANSAC: Learning Where to Sample Model Hypotheses”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2019, pp. 4322–4331.
- [15] Wieland Brendel, Jonas Rauber, and Matthias Bethge. “Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models”. In: *International Conference on Learning Representations (ICLR)*. 2018.

- [16] Tom B Brown et al. "Adversarial Patch". In: *arXiv preprint arXiv:1712.09665* (2017).
- [17] Nicholas Carlini and David Wagner. "Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods". In: *ACM Workshop on Artificial Intelligence and Security*. 2017, pp. 3–14.
- [18] Nicholas Carlini and David Wagner. "Towards Evaluating the Robustness of Neural Networks". In: *IEEE Symposium on Security and Privacy (SP)*. 2017, pp. 39–57.
- [19] Nicholas Carlini et al. "On Evaluating Adversarial Robustness". In: *arXiv preprint arXiv:1902.06705* (2019).
- [20] Vincent Casser et al. "Depth Prediction Without The Sensors: Leveraging Structure For Unsupervised Learning From Monocular Videos". In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2019, pp. 8001–8008.
- [21] Anirban Chakraborty et al. "Adversarial Attacks and Defences: A Survey". In: *arXiv preprint arXiv:1810.00069* (2018).
- [22] Haifeng Chen et al. "Robust Regression with Projection Based M-Estimators". In: *IEEE International Conference on Computer Vision (ICCV)*. 2003, pp. 878–885.
- [23] Tat jun Chin, Hanzi Wang, and David Suter. "The Ordered Residual Kernel for Robust Motion Subspace Clustering". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2009, pp. 333–341.
- [24] Tat-Jun Chin, Jin Yu, and David Suter. "Accelerated Hypothesis Generation for Multistructure Data via Preference Analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2012), pp. 625–638.
- [25] Tat-Jun Chin et al. "Efficient Globally Optimal Consensus Maximisation with Tree Search". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 2413–2421.
- [26] Niraj Chokshi. *New York Times: Is Alexa Listening? Amazon Echo Sent Out Recording of Couple's Conversation*. <https://www.nytimes.com/2018/05/25/business/amazon-alexa-conversation-shared-echo.html>. Accessed: 2020-01-05. 2018.
- [27] Thomas F Coleman and Yuying Li. "An Interior Trust Region Approach for Nonlinear Minimization Subject to Bounds". In: *SIAM Journal on optimization* (1996), pp. 418–445.
- [28] Dorin Comaniciu and Peter Meer. "Mean Shift: A Robust Approach Toward Feature Space Analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2002), pp. 603–619.
- [29] Andrew R Conn, Nicholas IM Gould, and Ph L Toint. *Trust Region Methods*. Vol. 1. SIAM, 2000.
- [30] Jifeng Dai et al. "R-fcn: Object detection via region-based fully convolutional networks". In: *arXiv preprint arXiv:1605.06409* (2016).
- [31] Nilaksh Das et al. "Keeping the Bad Guys Out: Protecting and Vaccinating Deep Learning with JPEG Compression". In: *arXiv preprint arXiv:1705.02900* (2017).
- [32] Andrew J Davison et al. "MonoSLAM: Real-Time Single Camera SLAM". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2007), pp. 1052–1067.
- [33] Jia Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 248–255.

- [34] Patrick Denis, James H Elder, and Francisco J Estrada. "Efficient Edge-Based Methods for Estimating Manhattan Frames in Urban Imagery". In: *European Conference on Computer Vision (ECCV)*. 2008, pp. 197–210.
- [35] Elvis Dohmatob. "Limitations of Adversarial Robustness: Strong no Free Lunch Theorem". In: *arXiv preprint arXiv:1810.04065* (2018).
- [36] David Eigen, Christian Puhersch, and Rob Fergus. "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014, pp. 2366–2374.
- [37] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct Sparse Odometry". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2017), pp. 611–625.
- [38] Jakob Engel, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-Scale Direct MonocularSLAM". In: *European Conference on Computer Vision (ECCV)*. 2014, pp. 834–849.
- [39] Logan Engstrom, Andrew Ilyas, and Anish Athalye. "Evaluating and Understanding the Robustness of Adversarial Logit Pairing". In: *arXiv preprint arXiv:1807.10272* (2018).
- [40] Logan Engstrom et al. "Exploring the Landscape of Spatial Robustness". In: *International Conference on Machine Learning (ICML)*. 2019, pp. 1802–1811.
- [41] Kevin Eykholt et al. "Robust Physical-World Attacks on Deep Learning Visual Classification". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 1625–1634.
- [42] Ronald Fagin, Ravi Kumar, and Dakshinamurthi Sivakumar. "Comparing Top-K Lists". In: *SIAM Journal on discrete mathematics* (2003), pp. 134–160.
- [43] Michela Farenzena, Andrea Fusiello, and Riccardo Gherardi. "Structure-and-Motion Pipeline on a Hierarchical Cluster Tree". In: *ICCV Workshops*. 2009, pp. 1489–1496.
- [44] Ethan Fetaya et al. "Understanding the Limitations of Conditional Generative Models". In: *International Conference on Learning Representations (ICLR)*. 2020.
- [45] M. A. Fischler and R. C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Communications of the ACM* (1981), pp. 381–395.
- [46] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. "SVO: Fast Semi-Direct Monocular Visual Odometry". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 15–22.
- [47] Victor Fragoso et al. "EVSAC: Accelerating Hypotheses Generation by Modeling Matching Scores with Extreme Value Theory". In: *IEEE International Conference on Computer Vision (ICCV)*. 2013, pp. 2472–2479.
- [48] Duncan P Frost, Olaf Kähler, and David W Murray. "Object-Aware Bundle Adjustment For Correcting Monocular Scale Drift". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 4770–4776.
- [49] Huan Fu et al. "Deep Ordinal Regression Network for Monocular Depth Estimation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 2002–2011.
- [50] Xiang Gao et al. "LDSO: Direct Sparse Odometry with Loop Closure". In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 2198–2204.

- [51] Ravi Garg et al. "Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue". In: *European Conference on Computer Vision (ECCV)*. 2016, pp. 740–756.
- [52] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 3354–3361.
- [53] Andreas Geiger, Julius Ziegler, and Christoph Stiller. "StereoScan: Dense 3D Reconstruction in Real-Time". In: *IEEE Intelligent Vehicles Symposium (IV)*. 2011, pp. 963–968.
- [54] Andreas Geiger et al. "Vision Meets Robotics: The KITTI Dataset". In: *IJRR* (2013).
- [55] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. "Unsupervised Monocular Depth Estimation with Left-Right Consistency". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 270–279.
- [56] Clément Godard et al. "Digging into self-Supervised Monocular Depth Estimation". In: *IEEE International Conference on Computer Vision (ICCV)*. 2019, pp. 3828–3838.
- [57] Kathrin Grosse et al. "On the (Statistical) Detection of Adversarial Examples". In: *arXiv preprint arXiv:1702.06280* (2017).
- [58] Michael Grupp. *evo: Python Package for the Evaluation of Odometry and SLAM*. <https://github.com/MichaelGrupp/evo>. 2017.
- [59] Chuan Guo et al. "Countering Adversarial Images using Input Transformations". In: *International Conference on Learning Representations (ICLR)* (2018).
- [60] Xiaoyang Guo et al. "Learning Monocular Depth by Distilling Cross-Domain Stereo Networks". In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 484–500.
- [61] Frank R Hampel et al. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley & Sons, 2011.
- [62] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge university press, 2003.
- [63] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034.
- [64] Geoffrey Hinton et al. "Deep Neural Networks for Acoustic Modeling in Speech Recognition". In: *IEEE Signal Processing Magazine* 29 (2012).
- [65] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial Transformer Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2015, pp. 2017–2025.
- [66] Anson Kahng et al. "Statistical Foundations of Virtual Democracy". In: *International Conference on Machine Learning (ICML)*. 2019, pp. 3173–3182.
- [67] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. "Adversarial Logit Pairing". In: *arXiv preprint arXiv:1803.06373* (2018).
- [68] Andrej Karpathy et al. "Large-Scale Video Classification with Convolutional Neural Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 1725–1732.
- [69] *Kernels*. [https://en.wikipedia.org/wiki/Kernel\\_\(statistics\)](https://en.wikipedia.org/wiki/Kernel_(statistics)). Accessed: 2020-09-20.
- [70] Georg Klein and David Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *International Symposium on Mixed and Augmented Reality*. 2007, pp. 1–10.

- [71] Maria Klodt and Andrea Vedaldi. "Supervising the New with the Old: Learning SFM from SFM". In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 698–713.
- [72] Florian Kluger et al. "CONSAC: Robust Multi-Model Fitting by Conditional Sample Consensus". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 4634–4643.
- [73] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2012, pp. 1097–1105.
- [74] Da Kuang, Sangwoon Yun, and Haesun Park. "SymNMF: Non-negative Low-Rank Approximation of a Similarity Matrix for Graph Clustering". In: *Journal of Global Optimization* (2015), pp. 545–574.
- [75] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial Machine Learning at Scale". In: *International Conference on Learning Representations (ICLR)* (2017).
- [76] Yevhen Kuznietsov, Jorg Stuckler, and Bastian Leibe. "Semi-Supervised Deep Learning for Monocular Depth Map Prediction". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6647–6655.
- [77] Taotao Lai et al. "A Unified Hypothesis Generation Framework for Multi-Structure Model Fitting". In: *Neurocomputing* (2017), pp. 144–154.
- [78] Taotao Lai et al. "Efficient Guided Hypothesis Generation for Multi-Structure Epipolar Geometry Estimation". In: *Computer Vision and Image Understanding* (2017), pp. 152–165.
- [79] Ruihao Li et al. "UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 7286–7291.
- [80] Xin Li and Fuxin Li. "Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics". In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5764–5772.
- [81] Yang Li, Yoshitaka Ushiku, and Tatsuya Harada. "Pose Graph Optimization for Unsupervised Monocular Visual Odometry". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5439–5445.
- [82] Yingzhen Li, John Bradshaw, and Yash Sharma. "Are Generative Classifiers More Robust to Adversarial Attacks?" In: *International Conference on Machine Learning (ICML)*. 2019, pp. 3804–3814.
- [83] Fangzhou Liao et al. "Defense against Adversarial Attacks Using High-Level Representation Guided Denoiser". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 1778–1787.
- [84] Fayao Liu et al. "Learning Depth From Single Monocular Images Using Deep Convolutional Neural Fields". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2015), pp. 2024–2039.
- [85] Jiacheng Liu and Ziyang Meng. "Visual SLAM With Drift-Free Rotation Estimation in Manhattan World". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6512–6519.
- [86] Shing Yan Loo et al. "CNN-SVO: Improving the Mapping in Semi-Direct Visual Odometry Using Single-Image Depth Prediction". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5218–5223.

- [87] David G Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision (IJCV)* (2004), pp. 91–110.
- [88] Chenxu Luo et al. “Every Pixel Counts++: Joint Learning of Geometry and Motion with 3D Holistic Understanding”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2019), pp. 2624–2641.
- [89] Ping Ma et al. “DSOD: DSO in Dynamic Environments”. In: *IEEE Access* (2019), pp. 178300–178309.
- [90] Aleksander Madry et al. <https://www.robust-ml.org/>. <https://www.robust-ml.org/>. Accessed: 2020-01-05.
- [91] Aleksander Madry et al. “Towards Deep Learning Models Resistant to Adversarial Attacks”. In: *International Conference on Learning Representations (ICLR)* (2018).
- [92] Luca Magri and Andrea Fusiello. “Multiple Model Fitting as a Set Coverage Problem”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3318–3326.
- [93] Luca Magri and Andrea Fusiello. “Robust Multiple Model Fitting with Preference Analysis and Low-rank Approximation.” In: *British Machine Vision Conference (BMVC)*. 2015, p. 12.
- [94] Luca Magri and Andrea Fusiello. “Scale Estimation in Multiple Models Fitting via Consensus Clustering”. In: *International Conference on Computer Analysis of Images and Patterns*. 2015, pp. 13–25.
- [95] Luca Magri and Andrea Fusiello. “T-Linkage: A Continuous Relaxation of J-Linkage for Multi-Model Fitting”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 3954–3961.
- [96] Reza Mahjourian, Martin Wicke, and Anelia Angelova. “Unsupervised Learning of Depth and Ego-Motion from Monocular Video using 3D Geometric Constraints”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 5667–5675.
- [97] Jan Hendrik Metzen et al. “On Detecting Adversarial Perturbations”. In: *International Conference on Learning Representations (ICLR)* (2017).
- [98] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. “Adversarial Training Methods for Semi-Supervised Text Classification”. In: 2017.
- [99] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. “DeepFool: DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2574–2582.
- [100] Seyed-Mohsen Moosavi-Dezfooli et al. “Universal Adversarial Perturbations”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1765–1773.
- [101] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. “ORB-SLAM: a Versatile and Accurate Monocular SLAM System”. In: *IEEE Transactions on Robotics* (2015), pp. 1147–1163.
- [102] Raul Mur-Artal and Juan D Tardós. “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras”. In: *IEEE Transactions on Robotics* (2017), pp. 1255–1262.
- [103] Muzammal Naseer, Salman Khan, and Fatih Porikli. “Indoor scene understanding in 2.5/3d for autonomous agents: A survey”. In: *IEEE Access* 7 (2018), pp. 1859–1887.



- [104] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. “DTAM: Dense Tracking and Mapping in Real-Time”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2011, pp. 2320–2327.
- [105] Tianyu Pang et al. “Rethinking Softmax Cross-Entropy Loss for Adversarial Robustness”. In: *International Conference on Learning Representations (ICLR)*. 2020.
- [106] Nicolas Papernot and Patrick McDaniel. “Extending Defensive Distillation”. In: *arXiv preprint arXiv:1705.05264* (2017).
- [107] Nicolas Papernot et al. “Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks”. In: *IEEE Symposium on Security and Privacy (SP)*. 2016, pp. 582–597.
- [108] Nicolas Papernot et al. “Practical Black-Box Attacks Against Machine Learning”. In: *Asia conference on computer and communications security*. 2017, pp. 506–519.
- [109] Ghosh Partha, Losalka Arpan, and Black Michael J. “Resisting Adversarial Attacks Using Gaussian Mixture Variational Autoencoders”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2019, pp. 541–548.
- [110] Trung T Pham et al. “The Random Cluster Model for Robust Geometric Fitting”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2014), pp. 1658–1671.
- [111] Sudeep Pillai, Rareş Ambruş, and Adrien Gaidon. “Superdepth: Self-Supervised, Super-Resolved Monocular Depth Estimation”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 9250–9256.
- [112] Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. “Learning Monocular Depth Estimation with Unsupervised Trinocular Assumptions”. In: *International Conference on 3D Vision (3DV)*. 2018, pp. 324–333.
- [113] Aaditya Prakash et al. “Deflecting Adversarial Attacks with Pixel Deflection”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 8571–8580.
- [114] Edward Raff et al. “Barrage of Random Transforms for Adversarially Robust Defense”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 6528–6537.
- [115] René Ranftl and Vladlen Koltun. “Deep Fundamental Matrix Estimation”. In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 284–299.
- [116] Anurag Ranjan et al. “Competitive Collaboration: Joint Unsupervised Learning of Depth, Camera Motion, Optical Flow and Motion Segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 12240–12249.
- [117] Jonas Rauber, Wieland Brendel, and Matthias Bethge. “Foolbox: A Python Toolbox to Benchmark the Robustness of Machine Learning Models”. In: *arXiv preprint arXiv:1707.04131* (2017).
- [118] Shaoqing Ren et al. “Faster R-CNN: towards real-time object detection with region proposal networks”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.6 (2016), pp. 1137–1149.
- [119] William JJ Rey. *Introduction to Robust and Quasi-Robust Statistical Methods*. Springer Science & Business Media, 2012.
- [120] Jörg Rothe. “Borda Count in Collective Decision Making: A Summary of Recent Results”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2019, pp. 9830–9836.

- [121] Peter J Rousseeuw and Annick M Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, 2005.
- [122] Lea Schönherr et al. “Adversarial Attacks Against Automatic Speech Recognition Systems via Psychoacoustic Hiding”. In: *arXiv preprint arXiv:1808.05665* (2018).
- [123] Lukas Schott et al. “Towards the First Adversarially Robust Neural Network Model on MNIST”. In: *International Conference on Learning Representations (ICLR)*. 2019.
- [124] Yash Sharma and Pin-Yu Chen. “Bypassing Feature Squeezing by Increasing Adversary Strength”. In: *arXiv preprint arXiv:1803.09868* (2018).
- [125] Tianwei Shen et al. “Beyond Photometric Loss for Self-Supervised Ego-Motion Estimation”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019.
- [126] Tianwei Shen et al. “Self-supervised learning of depth and motion under photometric inconsistency”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.
- [127] Lu Sheng et al. “Unsupervised Collaborative Learning of Keyframe Detection and Visual Odometry Towards Monocular Deep SLAM”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2019, pp. 4302–4311.
- [128] Richard Shin and Dawn Song. “JPEG-Resistant Adversarial Images”. In: *Advances in Neural Information Processing Systems (NeurIPS) Workshop on Machine Learning and Computer Security*. 2017.
- [129] Chawin Sitawarin et al. “DARTS: Deceiving Autonomous Cars with Toxic Signs”. In: *arXiv preprint arXiv:1802.06430* (2018).
- [130] Shiyu Song and Manmohan Chandraker. “Robust Scale Estimation in Real-Time Monocular SFM for Autonomous Driving”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 1566–1573.
- [131] Jürgen Sturm et al. “A benchmark for the Evaluation of RGB-DSLAM Systems”. In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*. 2012, pp. 573–580.
- [132] Christian Szegedy et al. “Intriguing Properties of Neural Networks”. In: *International Conference on Learning Representations (ICLR)* (2014).
- [133] Keisuke Tateno et al. “CNN-SLAM: Real-Time Dense MonocularSLAM with Learned Depth Prediction”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6243–6252.
- [134] Zachary Teed and Jia Deng. “DeepV2D: Video to Depth with Differentiable Structure from Motion”. In: *International Conference on Learning Representations (ICLR)* (2019).
- [135] Ruwan B Tennakoon et al. “Robust Model Fitting Using Higher Than Minimal Subset Sampling”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 2015, pp. 350–362.
- [136] Mariano Tepper and Guillermo Sapiro. “Fast L1-NMF for Multiple Parametric Model Estimation”. In: *arXiv preprint arXiv:1610.05712* (2016).
- [137] Mariano Tepper and Guillermo Sapiro. “Nonnegative Matrix Underapproximation for Robust Multiple Model Fitting”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2059–2067.
- [138] *TESLA Model S Crashed Into Tractor Trailer*. <https://www.tesla.com/blog/tragic-loss>. Accessed: 2020-08-07.

- [139] Lokender Tiwari and Saket Anand. "DGSAC: Density Guided Sampling and Consensus". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 974–982.
- [140] Lokender Tiwari and Saket Anand. "Fast Hypothesis Filtering for Multi-Structure Geometric Model fitting". In: *IEEE International Conference on Image Processing (ICIP)*. 2016, pp. 3728–3732.
- [141] Lokender Tiwari, Saket Anand, and Sushil Mittal. "Robust Multi-Model Fitting Using Density and Preference Analysis". In: *Asian Conference on Computer Vision (ACCV)*. 2016, pp. 308–323.
- [142] Lokender Tiwari et al. "Dissecting Deep Networks into an Ensemble of Generative Classifiers for Robust Predictions". In: *arXiv preprint arXiv:2006.10679* (2020).
- [143] Lokender Tiwari et al. "Pseudo RGB-D for Self-Improving Monocular SLAM and Depth Prediction". In: *European Conference on Computer Vision (ECCV)* (2020).
- [144] Roberto Toldo and Andrea Fusiello. "Robust Multiple Structures Estimation with J-Linkage". In: 2008, pp. 537–547.
- [145] Fabio Tosi et al. "Learning Monocular Depth Estimation Infusing Traditional Stereo Knowledge". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [146] Dimitris Tsipras et al. "Robustness May be at Odds with Accuracy". In: *International Conference on Learning Representations (ICLR)* (2018).
- [147] Jonathan Uesato et al. "Adversarial Risk and the Dangers of Evaluating Against Weak Attacks". In: *International Conference on Machine Learning (ICML)*. 2018, pp. 5025–5034.
- [148] Jill Van Newenhizen. "The Borda Method is Most Likely to Respect the Condorcet Principle". In: *Economic Theory* (1992), pp. 69–83.
- [149] Martin Velas et al. "CNN for IMU Assisted Odometry Estimation using Velocity Lidar". In: *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 2018, pp. 71–77.
- [150] Chaoyang Wang et al. "Learning Depth from Monocular videos using direct methods". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 2022–2030.
- [151] H. Wang, J. Cai, and J. Tang. "AMSAC: An Adaptive Robust Estimator for Model Fitting". In: *IEEE International Conference on Image Processing (ICIP)*. 2013, pp. 305–309.
- [152] H. Wang, T. J. Chin, and D. Suter. "Simultaneously Fitting and Segmenting Multiple-Structure Data with Outliers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2012), pp. 1177–1192.
- [153] Hanzi Wang and David Suter. "Robust Adaptive-Scale Parametric Model Estimation for Computer Vision". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2004), pp. 1459–1474.
- [154] Qinglong Wang et al. "Learning Adversary-Resistant Deep Neural Networks". In: *arXiv preprint arXiv:1612.01401* (2016).
- [155] Rui Wang, Stephen M Pizer, and Jan-Michael Frahm. "Recurrent Neural Network for Un-supervised Learning of Monocular Video Visual Odometry and Depth". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5555–5564.

- [156] Rui Wang, Martin Schworer, and Daniel Cremers. "Stereo DSO: Large-Scale Direct Dparse Visual Odometry with Stereo Cameras". In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3903–3911.
- [157] Sen Wang et al. "Deepvo: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 2043–2050.
- [158] Sen Wang et al. "End-to-end, Sequence-to-Sequence Probabilistic Visual Odometry through Deep Neural Networks". In: *IJRR* (2018), pp. 513–542.
- [159] Jamie Watson et al. "Self-Supervised Monocular Depth hints". In: *IEEE International Conference on Computer Vision (ICCV)*. 2019, pp. 2162–2171.
- [160] Hoi Sim Wong et al. "A Simultaneous Sample-and-Filter Strategy for Robust Multi-Structure Model Fitting". In: *Computer Vision and Image Understanding* (2013), pp. 1755–1769.
- [161] Hoi Sim Wong et al. "Dynamic and Hierarchical Multi-Structure Geometric Model Fitting". In: *IEEE International Conference on Computer Vision (ICCV)*. 2011, pp. 1044–1051.
- [162] Tong Wu, Liang Tong, and Yevgeniy Vorobeychik. "Defending Against Physically Realizable Attacks on Image Classification". In: *International Conference on Learning Representations (ICLR)* (2020).
- [163] Cihang Xie et al. "Adversarial examples for semantic segmentation and object detection". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1369–1378.
- [164] Cihang Xie et al. "Feature Denoising for Improving Adversarial Robustness". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 501–509.
- [165] Cihang Xie et al. "Mitigating Adversarial Effects Through Randomization". In: *International Conference on Learning Representations (ICLR)* (2018).
- [166] Lei Xu, Erkki Oja, and Pekka Kultanen. "A New Curve Detection Method: Randomized Hough Transform (RHT)". In: *Pattern Recognition Letters* (1990), pp. 331–338.
- [167] Weilin Xu, David Evans, and Yanjun Qi. "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks". In: *Network and Distributed Systems Security Symposium (NDSS)* (2018).
- [168] Fei Xue et al. "Beyond Tracking: Selecting Memory and Refining Poses for Deep Visual Odometry". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 8575–8583.
- [169] Fei Xue et al. "Guided Feature Selection for Deep Visual Odometry". In: *Asian Conference on Computer Vision (ACCV)*. 2018, pp. 293–308.
- [170] Nan Yang et al. "Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry". In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 817–833.
- [171] Zhenheng Yang et al. "LEGO: Learning Edge with Geometry all at once by Watching Videos". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 225–234.
- [172] Zhenheng Yang et al. "Unsupervised Learning of Geometry with Edge-Aware Depth-Normal Consistency". In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2018.

- [173] Jian Yao, Sanja Fidler, and Raquel Urtasun. "Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation". In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 702–709.
- [174] Zhewei Yao et al. "Trust Region Based Adversarial Attack on Neural Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 11350–11359.
- [175] Xiaochuan Yin et al. "Scale Recovery for Monocular Visual Odometry using Depth Estimated with Deep Convolutional Neural Fields". In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5870–5878.
- [176] Zhichao Yin and Jianping Shi. "GeoNet: Unsupervised Learning of dense Depth, optical flow and camera pose". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 1983–1992.
- [177] H Peyton Young. "Condorcet's Theory of Voting". In: *American Political science review* (1988), pp. 1231–1244.
- [178] Jin Yu, Tat-Jun Chin, and David Suter. "A Global Optimization Approach to Robust Multi-Model Fitting". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011, pp. 2041–2048.
- [179] Michał Zajac et al. "Adversarial Framing for Image and Video Classification". In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2019, pp. 10077–10078.
- [180] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. "Efficient Defenses Against Adversarial Attacks". In: *ACM Workshop on Artificial Intelligence and Security*. 2017, pp. 39–49.
- [181] Huangying Zhan et al. "Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 340–349.
- [182] Zhanyuan Zhang et al. "Clipped BagNet: Defending Against Sticker Attacks with Clipped Bag-of-features". In: *Deep Learning and Security Workshop (DLS)*. 2020.
- [183] Chen Zhao et al. "NM-Net: Mining Reliable Neighbors for Robust Feature Correspondences". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 215–224.
- [184] Stephan Zheng et al. "Improving the Robustness of Deep Neural Networks via Stability Training". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4480–4488.
- [185] Lipu Zhou and Michael Kaess. "Windowed Bundle Adjustment Framework for Unsupervised Learning of Monocular Depth Estimation With U-Net Extension and Clip Loss". In: *IEEE Robotics and Automation Letters* (2020), pp. 3283–3290.
- [186] Tinghui Zhou et al. "Unsupervised Learning of Depth and Ego-Motion from Video". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1851–1858.
- [187] Bingbing Zhuang et al. "Degeneracy in self-calibration revisited and a deep learning solution for uncalibrated slam". In: *arXiv preprint arXiv:1907.13185* (2019).
- [188] Bingbing Zhuang et al. "Learning structure-and-motion-aware rolling shutter correction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4551–4560.
- [189] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. "DF-Net: Unsupervised Joint Learning of Depth and Flow using Cross-Task Consistency". In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 36–53.

- 
- [190] Marco Zuliani, Charles S Kenney, and BS Manjunath. "The multiRANSAC Algorithm and its Application to Detect Planar Homographies". In: *IEEE International Conference on Image Processing (ICIP)*. 2005, pp. III-153.