Monocular Multibody SLAM in Dynamic Environments

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Engineering
by Research

by

GOKUL B. NAIR
201502034
`gokulb.nair@research.iiit.ac.in`

INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY
HYDERABAD

International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2020

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled **"Monocular Multibody SLAM in Dynamic Environments"** by **Gokul B. Nair**, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Prof. K. Madhava Krishna

To my family and friends

Acknowledgments

I would first like to express my feelings of gratitude to my research advisor Prof. K. Madhava Krishna at Robotics Research Centre (RRC). His prompt guidance and trust in my effort have been integral to the work that has led to this thesis. I must also thank my mentors J. Krishna Murthy, Junaid Ahmed Ansari, and Sarthak Sharma. They have been very patient with me throughout my time here and I have learned an immense lot under their mentorship. I must also thank Bharat Gopalakrishnan, whose timely discussions and advice have helped me greatly at crucial junctures of my work.

I am also grateful to my co-authors Swapnil Daga, Rahul Sajnani, and Anirudha Ramesh. I couldn't have asked for a more cooperative team to work with. I also had the privilege of having a delightful company at the lab, thanks to Josyula Gopala Krishna, Jyotish P., Unnikrishnan R. Nair, and Nivedita Rufus. They have always been open to help me out when needed. I have been blessed with a huge number of wonderful friends who have accompanied me during my time here at IIIT-H. Their friendship and support have driven me through my ups and downs while here.

Finally, I must thank my parents and my brother for their constant support, encouragement, prayers, and trust in my efforts.

Abstract

This thesis proposes a novel pose-graph based formulation to map moving object trajectories onto a stationary global reference frame where the moving objects are observed from a single moving camera. This is performed in a *multi-body* fashion as a consequence of which we obtain accurate estimations to both the ego-camera trajectories as well as trajectories to multiple dynamic participants in the scene. This set up is dovetailed to on-road dynamic vehicles observed from a monocular sensor mounted on a moving ego-car such as in the KITTI dataset. The original triangulation problem is *intractable* in this scenario as it is impossible to triangulate a moving object from a single moving camera *unambiguously* unless there are appropriate restrictions on the object motion. The problem of *unobservability* also manifests in the form of *the relative scale problem* in classical multi-body SFM/SLAM formulations wherein there exists an unresolved scale factor (called as relative scale) such that a family of infinitely many solutions exist for any pair of motions in the scene. This prevents accurately representing the moving vehicle and camera in the same stationary frame. We overcome this relative scale factor by leveraging single-view metrology, advances in deep learning, and category-level shape estimation.

We show in this paper by invoking single view reconstruction techniques, that moving objects can be represented in the same global frame in which the camera trajectory is represented without any assumptions or restrictions on object motion. More specifically we solve for the relative scale problem through a factor graph formulation where the nodes include camera and moving object poses and thereby obtain trajectories of the moving object in the starting camera frame. We use Gaussian Process based motion model prediction and lane constraints to further improve the trajectory estimates and show performance gain with previous formulations that have attacked this problem. This optimization helps us reduce the average error in trajectories of multiple bodies over real-world datasets, such as KITTI. To the best of our knowledge, our method is the first *practical* monocular multi-body SLAM system to perform *dynamic* multi-object and ego localization in a *unified framework* in *metric* scale.

Contents

Chapter Page

1 Introduction . . . 1
1.1 Related Works . . . 2
1.2 Key Contributions . . . 4
1.3 Thesis Organization . . . 4

2 Background . . . 6
2.1 Pose-Shape Adjustments for Object Localization . . . 6
2.1.1 Shape Prior . . . 6
2.1.2 Pose and Shape Adjustments . . . 7
2.2 g2o: Hyper-Graph based Optimization Library . . . 7
2.2.1 Elements . . . 7

3 Multi-Object Monocular SLAM in Dynamic Environments . . . 10
3.1 Overview of the Proposed Pipeline . . . 10
3.2 Vehicle Localization and Odometry Estimation . . . 11
3.2.1 Depth Estimation for Points on Ground Plane . . . 11
3.2.2 Odometry Estimations . . . 11
3.2.2.1 Scale Ambiguous Odometry . . . 11
3.2.2.2 Scale Initialization . . . 12
3.2.3 Pose and Shape Adjustments Pipeline . . . 12
3.3 Multi-Object Pose Graph Optimizer . . . 13
3.3.1 Pose-Graph Formulation . . . 13
3.3.2 Cost Functions . . . 14
3.3.3 Confidence Parameterization . . . 15
3.4 Experiments and Results . . . 16
3.4.1 Dataset . . . 16
3.4.2 Approaches Evaluated . . . 17
3.4.3 Qualitative Results . . . 17
3.4.3.1 Pose and Shape Adjustments . . . 17
3.4.3.2 Odometry Estimation . . . 17
3.4.3.3 Pose-Graph Optimization . . . 18
3.4.4 Quantitative Results . . . 21
3.4.4.1 Odometry Estimations . . . 21
3.4.4.2 Pose-Graph Optimization . . . 21
3.4.5 Summary of Results . . . 23

4 Mapping Trajectories of On-Road Dynamic Participants with a Moving Monocular Camera: A Multibody Approach 24
4.1 Overview of the Proposed Pipeline 24
4.1.1 Vehicle Localization 24
4.1.2 Visual Odometry 24
4.1.3 Trajectory Prediction 26
4.1.4 Pose-Graph Optimizer 26
4.2 Obtaining Trajectory Initializations in Global Frame 26
4.2.1 3D Depth Estimator 26
4.2.2 3D Object Localization from Pose-Shape Optimizer 27
4.2.3 Ego Trajectory Estimation and Scale Initializer 27
4.3 Gaussian Processing(GP) based Vehicle Trajectory Prediction 28
4.4 Pose-Graph Formulation 30
4.4.1 Binary-Edges 30
4.4.2 Unary-Edges 30
4.4.3 Confidence Parameterization 30
4.5 Exploiting Dynamic cues in Pose-Graph Optimizer 31
4.6 Exploiting Static Cues from the Environment 32
4.7 Cumulative Pose-Graph Formulation 33
4.8 Experiments and Results 33
4.8.1 System Setup 33
4.8.2 Dataset 34
4.8.3 Approaches Evaluated 34
4.8.4 Experiments 34
4.8.4.1 Real-Time Approach with Scale Initialized Odometry 34
4.8.4.2 Batch-Wise Approach with Scale Ambiguous Odometry 35
4.8.5 Qualitative Results 35
4.8.5.1 Batch-Wise Version with Scale Ambiguous Odometry 36
4.8.5.2 Real-Time Version with Scale Initialized Odometry 36
4.8.6 Quantitative Results 36
4.8.7 Ablation Studies 37
4.8.7.1 Impact of Lane-based Constraints in Pose-Graph Formulation 37
4.8.7.2 Impact of Depth-Thresholding in Lane-Constraints 43
4.8.7.3 Run-time Analysis for the Real-Time Approach with Scale Initialized Odometry 44

5 Conclusions 46

Bibliography 48

List of Figures

Figure Page

1.1 Image illustrates the intractability in obtaining reasonable reconstructions for an object. Three instances of the camera and an object are shown at consecutive time instants. Clearly, any attempt to triangulate results in the reconstructions showing up at wrong points shown as red points. 1

1.2 Part (b) illustrates the *relative scale* problem where the actual path between the black terminal points is represented by the green trajectory, which is in say *metric scale*. The estimated trajectory for the same path may however be obtained as any of the red trajectories which are similar to the required trajectory but off by a single parameter scale factor. These red trajectories are said to be in ambiguous scale. 2

1.3 Part (a) illustrates the the scene being captured by the monocular sensor set up along with bounding box detection shown i nthe form of red, green and blue boxes for dynamic vehicles in scene. Part (b) illustrates the mapping that we aim to obtain through the work detailed by this thesis. The red, green blue trajectories along with the 3D show the estimated mapping for the detected vehicles in Part (a) with bounding boxes of the same colour. The black trajectory and wireframe represents the estimate for he ego-camera motion. This plot ilustrates how we intend to obtain accurate estimates for the ego-camera as well as various dynamic vehicles in it's scene in a stationary *global frame* in a *unified metric scale*. 5

2.1 Illustration of the *shape prior*. The red wireframe illustrates the set of initial 3D mean shape which consists of k ordered keypoints that are visible in the wireframe in the form of $k = 36$ corners and wheel centres respectively. This wireframe is deformed to fit to the specifications of various vehicle instances found in the camera's scene using a set of B vectors. Th blue wireframe shows an example of one such instance of vehicle structure that the wireframe attempts to fit to with help from the scaling of the deformation vectors represented using the black directed arrows. 6

2.2 Illustration of the pose-graph elements we make use in this thesis. The left most illustration is that of 'A' node with its parameterization T_A^W. The central illustration is that of a binary-edge from a node 'A' with *estimate* T_A^W to a node 'B' with *estimate* T_B^W. The edge *measurement* is T_B^A. The rightmost illustration is that of a unary edge from a node 'A' with *estimate* T_A^W to static environment point 'p'. The *measurement* of the unary-edge is given as $\mathbf{tr}_p^W$ 8

3.1 **Pipeline**: We obtain dynamic-vehicle localizations via the modules explained in blue section. The mathematical representations to the same can be found in Sec.3.2.3 and Sec.3.2.1. The green section illustrates our approach to obtain accurate odometry estimations in metric scale, as explained in Sec.3.2.2. The orange section illustrates a part of the pose-graph structure where the gray, the orange and the purple nodes represent the nodes for ego-car and two dynamic vehicles in the scene respectively. Moreover, the black, the blue and the red edges represent the camera-camera, vehicle-vehicle and camera-vehicle edges respectively. 11

3.2 Illustration of a simple pose-graph defined by a constraint defined from nodes A to B by a binary edge. 13

3.3 An illustration of the multi-body pose graph structure under our setting between a pair of consecutive frames. Nodes in blue correspond to the primary pose graph i.e., pose graph for the ego-motion. Nodes in green correspond to the secondary pose graph i.e., pose graph for the dynamic objects in the scene. 14

3.4 Localizations in ego-camera frame after pose and shape estimations for a dynamic multi-vehicle scenario. 18

3.5 Odometry estimations in metric scale in blue. GPS/IMU trajectory is in red and ORB trajectory in its scale is in yellow. The figure illustrates that our odometry estimation is proficient on sharp turns and long sequences. 18

3.6 Qualitative results on various sequences for **Batch-Wise Approach with Scale Initialized Odometry**. *Col 1* shows input images with bounding boxes for vehicles mapped in *Col 2* and *Col 3*. While *Row 1* and *Row 3* illustrate our performance on multi-vehicle traffic scenarios, *Row 2* shows results for a far away vehicle over a long sequence. Ego-vehicle is shown in black whereas the red, blue and green plots show unique dynamic vehicles in the scene with corresponding dotted plots showing the ground truths. Note that entire ground truth trajectory is shown at once in the figures whereas our results are up to the instance frame shown in *Col 1*. 19

3.7 Qualitative top-view plots for **Batch-Wise Approach with Scale Initialized Odometry**. The three plots show the top view version for the same sequences shown in the perspective 3D mapping illustrated in Fig.3.6. The road plane is represented in cyan whereas the odometry estimates are show as a black trajectory. Other dynamic vehicles in the camera's scene are show as red, green and blue trajectories. Dotted trajectories represent the respective ground truth trajectories. 20

4.1 Detailed Illustration of the proposed pipeline. 25

4.2 An illustration of the pose-graph formulation that resolves *cyclic-consistency* between object and camera across consecutive images. Triangular blocks represent ego-camera while rectangular blocks represent dynamic objects in the ego-camera's scene respectively. Red, green and black edges represent camera-camera, camera-object and object-object edges respectively. 32

4.3 An illustration of lane-based pose-graph formulation from Sec. 4.6. Triangle blocks represent agent instances, directed red edges represent agent-agent biary-edges and undirected blue edges represent unary-edges between the agent instance and the stationary lane-points in environment. 33

4.4 Performance of all pipelines evaluated as summarized in Sec.3.4.2 for two vehicle instances which are Part(a): Sequence 18, Car ID 2 and Part (b): Sequence 19, Ego Car. While blue and yellow trajectories represent trajectory estimates from CubeSLAM[47] and Namdev*et al.*[25] respectively; red, violet and green trajectories illustrated the performance of our three experiments from Chapter.3, Sec.4.8.4.1 and Sec.4.8.4.2 respectively. Ground truth trajectory is plotted as a light blue trajectory. 37

4.5 Qualitative results on various sequences for **Batch-Wise Approach with Scale Ambiguous Odometry** as explained in Sec.4.8.4.2. *Col 1* shows input images with bounding boxes for vehicles mapped in *Col 2* and *Col 3*. While *Row 1* and *Row 3* illustrate our performance on multi-vehicle traffic scenarios, *Row 2* shows results for a far away vehicle over a long sequence. Ego-vehicle is shown in black whereas the red, blue and green plots show unique vehicles in the scene with corresponding dotted plots showing the ground truths. Note that entire ground truth trajectory is shown at once in the figures whereas our results are up to the instance frame shown in *Col 1*. 38

4.6 Qualitative top-view plots for **Batch-Wise Approach with Scale Ambiguous Odometry** from Sec.4.8.4.2. The three plots show the top view version for the same sequences shown in the perspective 3D mapping illustrated in Fig.3.6. The road plane is represented light red, light green and light violet planes whereas the odometry estimates are show as a black trajectory. Other dynamic vehicles in the camera's scene are show as red, green and blue trajectories. Dotted trajectories represent the respective ground truth trajectories. 39

4.7 Qualitative results on various sequences for **Real-Time Approach with Scale Initialized Odometry** as described in Sec.4.8.4.1. *Col 1* shows input images with bounding boxes for vehicles mapped in *Col 2* and *Col 3*. While *Row 1* and *Row 3* illustrate our performance on multi-vehicle traffic scenarios, *Row 2* shows results for a far away vehicle over a long sequence. Ego-vehicle is shown in black whereas the red, blue and green plots show unique vehicles in the scene with corresponding dotted plots showing the ground truths. Note that entire ground truth trajectory is shown at once in the figures whereas our results are up to the instance frame shown in *Col 1*. 40

4.8 Qualitative top-view plots for **Real-Time Approach with Scale Initialized Odometry** from Sec.4.8.4.1. The three plots show the top view version for the same sequences shown in the perspective 3D mapping illustrated in Fig.3.6. The road plane is represented as light red, light green an light violet planes whereas the odometry estimates are show as a black trajectory. Other dynamic vehicles in the camera's scene are show as red, green and blue trajectories. Dotted trajectories represent the respective ground truth trajectories. 41

4.9 Plot illustrating how number of objects in scene do not affect the time-elapsed in our optimization formulation from Sec. 4.8.4.1 45

List of Tables

Table Page

3.1 Analysis between various threshold settings for odometry estimations by computing Absolute Translation Error (ATE) in metres. 21

3.2 ATE for all dynamic vehicles in the ego-camera's scene in a static-global frame recognized by the formulation across various sequences. The tables considers the ATEs obtained for the experiments when various sets of edges are taken into consideration by the optimization problem. 22

3.3 ATE for all ego-trajectory estimates in a static-global frame recognized by the formulation across various sequences. The tables considers the ATEs obtained for the experiments when various sets of edges are taken into consideration by the optimization problem. 22

3.4 Quantitative performance for various dynamic vehicles in the scene of the ego-camera for all evaluated approaches as summarized in Sec.3.4.2 in a static-global frame in a single *metric scale*. The errors are computed as root-mean-squared Absolute Translation Errors averaged throughout across complete trajectories for each of the vehicles instances reported in the table. 23

3.5 Quantitative performance for ego-camera trajectories in the scene for all evaluated approaches as summarized in Sec.3.4.2 in a static-global frame in a single *metric scale*. The errors are computed as root-mean-squared Absolute Translation Errors averaged throughout across complete trajectories for each of the vehicles instances reported in the table. 23

4.1 Quantitative performance for various dynamic vehicles in the scene of the ego-camera for all evaluated approaches as summarized in Sec.3.4.2 in a static-global frame in a single *metric scale*. The errors are computed as root-mean-squared Absolute Translation Errors averaged throughout across complete trajectories for each of the vehicles instances reported in the table. 42

4.2 Quantitative performance for ego-camera trajectories in the scene for all evaluated approaches as summarized in Sec.3.4.2 in a static-global frame in a single *metric scale*. The errors are computed as root-mean-squared Absolute Translation Errors averaged throughout across complete trajectories for each of the vehicles instances reported in the table. 42

4.3 Ablation studies on impact of lane-based constraints on batch-based approach from Sec.4.8.4.2 for trajectories of various instances of dynamic vehicles in the ego-camera's scene across various recorded sequences from our evaluation data. 43

4.4 Ablation studies on impact of lane-based constraints on batch-based approach from Sec.4.8.4.2 for various instances of ego-camera trajectories across various recorded sequences from our evaluation data.. 43
4.5 Analysis over depth parameter τ for lane-based optimization formulation in batch-based approach from Sec. 4.6 for trajectories of various instances of dynamic vehicles in the ego-camera's scene across various recorded sequences from our evaluation data. 44
4.6 Analysis over depth parameter τ for lane-based optimization formulation in batch-based approach from Sec. 4.6 for various instances of ego-camera trajectories across various recorded sequences from our evaluation data. 44

Chapter 1

Introduction

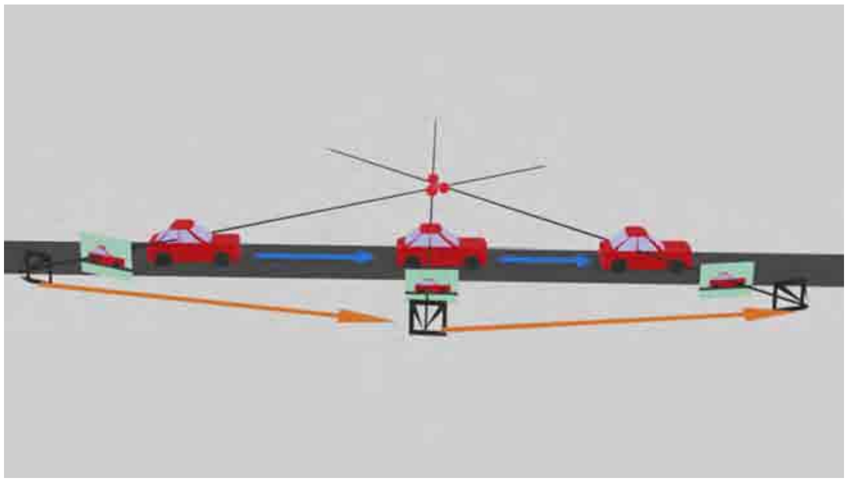


Figure 1.1: Image illustrates the intractability in obtaining reasonable reconstructions for an object. Three instances of the camera and an object are shown at consecutive time instants. Clearly, any attempt to triangulate results in the reconstructions showing up at wrong points shown as red points.

Autonomous driving has been a topic of interest to a wide array of stakeholders coming from both commercial and academic backgrounds. This has led to a heavy investment of time and effort into developing safe and robust solutions to various problems posed for its success. One of the crucial aspects that contribute to autonomous driving is its ability to make intelligent observations from visual cues obtained from a constantly changing environment around it in traffic scenarios. Extensive work in simultaneous localization and mapping (SLAM) has led researchers to achieve effective solutions to tackle this problem while utilizing long-range perception capabilities of sensors such as LiDAR, radar and stereo camera setup.

A widespread ambition to commercialize autonomous vehicles has inspired rigorous exploration of perception using more economic sensors like single-camera setup. Consequently, research in monocular SLAM has matured significantly over the last few decades, resulting in very stable *off-the-shelf* solutions [7, 9, 21]. However, dynamic scenes still pose unique challenges for even the best such solu-

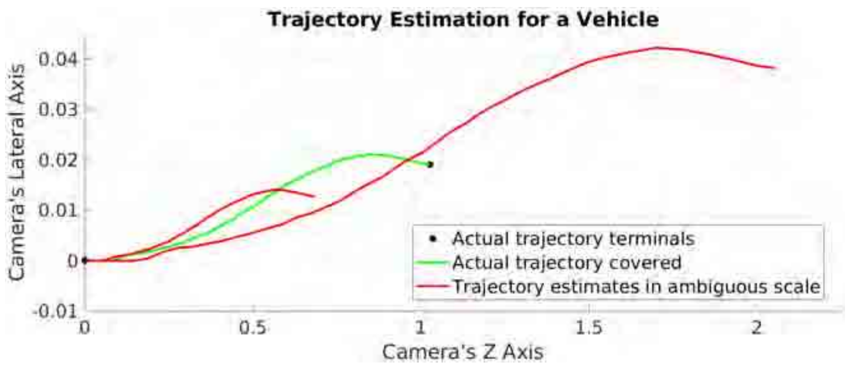


Figure 1.2: Part (b) illustrates the *relative scale* problem where the actual path between the black terminal points is represented by the green trajectory, which is in say *metric scale*. The estimated trajectory for the same path may however be obtained as any of the red trajectories which are similar to the required trajectory but off by a single parameter scale factor. These red trajectories are said to be in ambiguous scale.

tions. As illustrated in figure 1.1, any attempt to triangulate a moving object from temporally separated ego-camera-poses results in erroneous estimates rendering the original problem intractable. Traditionally, formulations prior to the Deep Learning era overcame this by introducing restrictions to motions estimates of any or both of the ego-camera or vehicles in scene [17, 25, 27]. However, these restrictions can now be relaxed due to effective depth, pose and shape estimation capabilities in single view [15, 24, 31, 41, 48].

In this thesis, we tackle a more general version of the monocular SLAM problem in dynamic environments: ***multi-body visual SLAM***. While monocular SLAM methods traditionally track only the ego-camera motion, multi-body SLAM deals with the *explicit* pose estimation of multiple dynamic objects (dynamic *bodies*) as well. Despite being an extremely useful problem, it has not received comparable attention to its *uni-body* counterpart (i.e., SLAM using stationary *landmarks*). This can primarily be attributed to the *ill-posedness* of monocular multibody Structure-from-Motion [27]. While the scale factor ambiguity of monocular SLAM is well-known [6, 7, 9, 21], the lesser-known-yet-well-studied *relative scale* ambiguity persists with multibody monocular SLAM [17, 25, 27, 28, 33, 42, 44]. This has been illustrated in detail in figure 1.2.

1.1 Related Works

As mentioned before Simultaneous Localization and Mapping (SLAM) is a topic well sought after by the researchers around the world. As a result, there have been a huge number of well received approaches that tackle this problem and showcase commendable performances using sensors like LiDAR, stereo as well as monocular setup. Given that we strictly stick to a single camera for a sensor to tackle this problem, we limit our survey of related works also to the approaches that utilize the monocular setup.

Classical multibody frameworks with monocular camera have approached the problem along two paradigms of thought. In the first, the camera poses are assumed to be known even as the moving object gets triangulated from temporally separated poses. The observability problem manifests in that such moving object triangulation necessarily imposes restrictions on object and or camera motion [29, 46]. The earliest approaches to monocular multibody SLAM [5, 8, 13, 19, 44] were based on *motion segmentation*: segmenting multiple motions from a set of triangulated points. Extending epipolar geometry to multiple objects, *multibody fundamental matrices* were used in [19, 33, 42, 44].

Trajectory triangulation methods [3, 28], on the other hand, derive a set of constraints for trajectories of objects, and solve the multi-body SLAM problem under these constraints. In the other paradigm, segmented moving objects are individually subject to the SLAM process. This gives a 3D rendering of the object as though they are stationary along with a camera trajectory relative to those objects. A similar SLAM treatment applied to the stationary world devoid of moving objects gives the *camera trajectory with respect to the stationary world* along with the *reconstruction of the stationary world.* Unification of these representations into the stationary frame involves solving the difficult relative scale problem. Here again, the ill-posed nature manifests in that the relative scale can only be computed by imposing assumptions on camera and/or object motion [27, 25]. There have been many other works [45, 4] that treat the SLAM problem in dynamic scenes by removing the dynamic participants.

Ozden *et al.* [27] extend the multi-body Structure-from-Motion framework [8] to cope with practical issues, such as a varying number of independently moving objects, track failure, etc. Another class of approaches applies model selection methods to segment independently moving objects in a scene, and then explicitly solve for relative scale solutions [42, 8, 5]. It is worth noting that the above approaches operate *of ine*, and extending them for online operation is non-trivial.

Kundu *et al.* [17] proposed a fast, incremental multi-body SLAM system that leverages motion segmentation to assign feature tracks to dynamic bodies, and then independently for relative-scale for the segmented motions. Critical to their success is the underlying assumption of smooth camera motions. Later Namdev *et al.* [25] provided analytical solutions for a restricted set of vehicle motions, (linear, planar, and non-holonomic). The underlying principle here is that since the problem of computing relative scale is fundamentally ill-posed it can be computed with restrictions on motions of the dynamic participants in the scene. More recently, Ranftl *et al.* [32] presented a dense monocular depth estimation pipeline targeted at dynamic scenes, and resolve *relative scale* ambiguity. Using optical ow and motion segmentation, they formulate a convex optimization problem to jointly estimate pixel correspondences across two-views. However, the obtained solution is still scale-ambiguous.

With the advent of deep learning With the advent of Deep Learning and the subsequent rich representations of objects in metric scale from a single view, some of the restrictive assumptions entailed in classical frameworks can now be relaxed. Improvements to object detection [40, 14, 11, 38] and motion segmentation have resulted in such methods directly being employed in multi-body SLAM. Reddy *et al.* [36] and Li *et al.* [18] present approaches to multi-body SLAM using a stereo camera. Here however, the problem is *observable*, while we handle the harder *unobservable* case of monocular cameras.

In this thesis, we showcase trajectories of dynamic participants mapped in the same reference frame as ego trajectories. We distinguish this work from the classical formulations by its lack of any restrictions on the vehicle and camera trajectories even as we showcase significant performance gain in terms of ATE(Absolute Translation Error) metric vis a vis classical formulations [25]. We also contrast ourselves with modern single view and multi view object reconstruction frameworks [24, 23, 2, 6, 31, 20] in that we go beyond representing objects only in the local camera frame. We also show performance gain vis a vis very recent frameworks such as CubeSLAM[47] that have tried to use object representations to improve ego-vehicle trajectories even as we go past [47] by accurately mapping and benchmarking trajectories of dynamic participants.

Other related works include 3D object representation from images in single and over multiple views wherein linear and non-linear object priors are used for accurate localization in 3D[24, 31], those that integrate objects into a SLAM framework indoors [30] as well as outdoors [47], and those that use object representations to resolve difficult geometry problems such as scale drift[6].

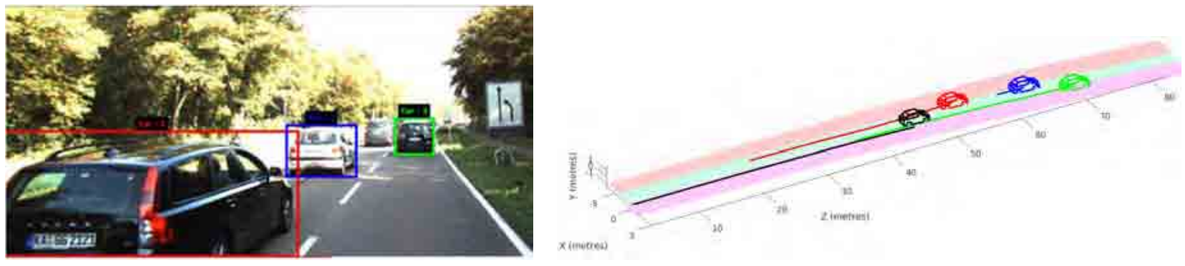
1.2 Key Contributions

We propose a factor graph formulation over ego-camera and single view object poses and demonstrate its ability to solve for all trajectories *unambiguously* in a multi-body fashion to provide resultant estimates in a *unified global frame* and in a *single metric scale*[1]. This pose-graph formulation takes into account the inter-object relationships between the ego-camera and the various dynamic vehicles in it is scene in the form of the respective relative pose estimations obtained from multiple vehicle localization pipelines in the ego-camera frame. This is further enhanced by bringing in additional constraints into the pose-graph through lane priors and Gaussian Process based motion models. This module of the pose-graph formulation takes into account the benefits of taking the stationary environment features into the optimization formulation. We show high fidelity estimates of camera trajectory as well as those of other dynamic participants on road. Appropriate presentation of our results and ablation studies signify the proposed pipeline's competitiveness and the role of various key modules in it. To the best of our knowledge, this is the first monocular multi-body SLAM to represent moving obstacle trajectories in a *unified global metric frame*, on long real-world dynamic trajectories.

1.3 Thesis Organization

The entirety of our work has been categorized into three chapters broadly. Chapter.2 re-introduces some of the prerequisite works and provides a formal introduction to the hyper graph based optimization toolbox used for major contributions made through this thesis. Chapter.3 details our efforts and observations obtained from our work compiled in the form of our first listed publication in Chapter.5.

[1]We use the term *metric scale* to denote a coordinate frame in which all distances are expressed in units of metres



(a) (b)

Figure 1.3: Part (a) illustrates the the scene being captured by the monocular sensor set up along with bounding box detection shown i nthe form of red, green and blue boxes for dynamic vehicles in scene. Part (b) illustrates the mapping that we aim to obtain through the work detailed by this thesis. The red, green blue trajectories along with the 3D show the estimated mapping for the detected vehicles in Part (a) with bounding boxes of the same colour. The black trajectory and wireframe represents the estimate for he ego-camera motion. This plot ilustrates how we intend to obtain accurate estimates for the ego-camera as well as various dynamic vehicles in it's scene in a stationary *global frame* in a *unified metric scale*.

Similarly, Chapter.4 describes the ideas, experiments, observations and analyses prepared through our work compiled as our second paper listed in Chapter.5. We arrange the contents for these chapters in the form of various modules. They begin with an overview of the pipeline proposed through the respective works and follow into the this may involve the initialization pipelines that provide initializations to various essential components of the pose-graph formulations that we experiment with.

Chapter 2

Background

2.1 Pose-Shape Adjustments for Object Localization

2.1.1 Shape Prior

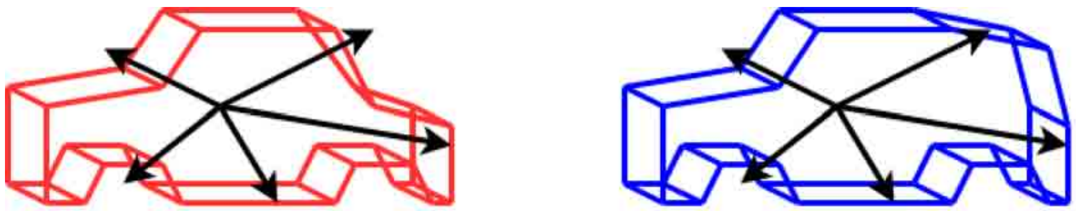


Figure 2.1: Illustration of the *shape prior*. The red wireframe illustrates the set of initial 3D mean shape which consists of k ordered keypoints that are visible in the wireframe in the form of $k = 36$ corners and wheel centres respectively. This wireframe is deformed to fit to the specifications of various vehicle instances found in the camera's scene using a set of B vectors. Th blue wireframe shows an example of one such instance of vehicle structure that the wireframe attempts to fit to with help from the scaling of the deformation vectors represented using the black directed arrows.

Borrowing the notations from Murthy *et al.*[24], we begin with a basis *shape prior* consisting of $k \quad \mathbb{Z} - *$ ordered keypoints, for the object used as a mean shape $\overline{X} \quad \mathbb{R}^{3k}$. Let $B \quad \mathbb{Z} - *$ basis vectors be $V \quad \mathbb{R}^{3k \times B}$ and the corresponding deformation coefficients be $\Lambda \quad \mathbb{R}^{B}$. Assuming that a particular object instance has a rotation of $R \quad SO(3)$ and translation of $\mathbf{tr} \quad \mathbb{R}^3$ with respect to the camera, its instance $X \quad \mathbb{R}^{3k}$ in the scene can be shown mathematically using the following *shape prior* model:

$$X = \hat{R} \times (\overline{X} + V \times \Lambda) + \hat{\mathbf{tr}} \tag{2.1}$$

Here, $\hat{R} = diag([R, R, R, ..., R]) \quad \mathbb{R}^{3k \times 3k}$ and $\hat{\mathbf{tr}} = (\mathbf{tr}^T, \mathbf{tr}^T, \mathbf{tr}^T, ..., \mathbf{tr}^T)^T \quad \mathbb{R}^{3k}$. Also, $\overline{X} = (\overline{X}_1^T, \overline{X}_2^T, \overline{X}_3^T, ..., \overline{X}_K^T) \quad \mathbb{R}^{3k}$ represents the basis *shape prior* and the resultant shape for the object instance is $X = (X_1^T, X_2^T, X_3^T, ..., X_k^T) \quad \mathbb{R}^{3k}$ where each $X_i \quad \mathbb{R}^3$ represents one of the k = 36 keypoints in 3D coordinate system from camera's perspective.

2.1.2 Pose and Shape Adjustments

Now, Let the ordered collection of keypoints in 2D image space be $\hat{x} = (\hat{x}_1^T, \hat{x}_2^T, \hat{x}_3^T, ..., \hat{x}_k^T) \quad \mathbb{R}^{2k}$. Given that π_k represents the function to project 3D coordinates onto 2D image space using the camera intrinsic parameters $\mu = (f_x, f_y, c_x, c_y)$, fairly accurate estimates for the pose parameters (R, tr) and the shape parameter (Λ) for the object instance can be obtained using the following objective function:

$$\min_{R,t,\Lambda} \mathcal{L}_r = \left\| \pi_k(\hat{R} \times (\overline{X} + V \times \Lambda) + \hat{\mathbf{tr}}; f_x, f_y, c_x, c_y) - \hat{x} \right\|_2^2 \tag{2.2}$$

$$\pi([X, Y, Z]^T, \mu) = \begin{pmatrix} \frac{f_x X}{Z} + c_x \\ \frac{f_y Y}{Z} + c_y \end{pmatrix} = \pi_K((X_1^T, ..., X_K^T)^T, \mu) = (\pi(X_1^T, \mu)^T, ..., \pi(X_K^T, \mu)^T)^T \tag{2.3}$$

Minimizing the objective function (*cf.* Eqn 2.2) separately for *pose parameters* (R, $\approx\diagdown$) and *shape parameters* (Λ) provides us with an optimal fitting of the *shape prior* over the dynamic object. We obtain the object orientation as R after *pose parameter adjustments*. The object's 3D coordinates from the camera $\mathtt{tr}$ is obtained from the mean of wheel centres.

2.2 g2o: Hyper-Graph based Optimization Library

A major part of the work as a part of the contributions made by this thesis has been performed on the g2o[16] library. It is a non-linear least squares based optimization framework[12] implemented in C++. The optimization problem definition is made by representing the initializations and the various constraints in the form of a hyper-graph formulation where various kinds of parameterizations of the many elements of the graph structure gives a mathematical meaning to the graph structure itself. It is called a pose-graph optimization framework because the main parameterization of any element of the pose-graph structure is a *pose* itself which may be defined relative to a generalized stationary world coordinate system or with respect to the coordinate frame of other objects in the scene which may also be represented as certain elements of the pose-graph formulation.

2.2.1 Elements

The g2o library allows various kinds of elements that can be incorporated in the pose-graph structure. Some of the major elements that we make use of in our pose-graph formulations in Sec.3.3.1 and Sec.4.4. Some of these elements are as follows:

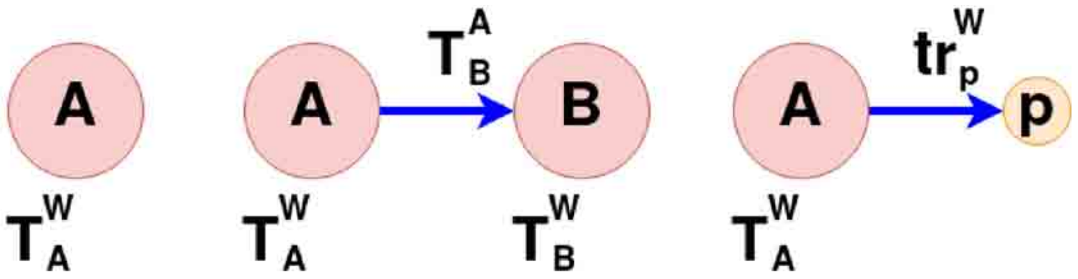


Figure 2.2: Illustration of the pose-graph elements we make use in this thesis. The left most illustration is that of 'A' node with its parameterization T_A^W. The central illustration is that of a binary-edge from a node 'A' with *estimate* T_A^W to a node 'B' with *estimate* T_B^W. The edge *measurement* is T_B^A. The rightmost illustration is that of a unary edge from a node 'A' with *estimate* T_A^W to static environment point 'p'. The *measurement* of the unary-edge is given as $\mathbf{tr}_p^W$

.

- **Node:** A node is often used to denote a separate local coordinate system which could be a whole object or a movable joint of an object. In our formulation, we utilize nodes to represent coordinate frames of each vehicle instances across various time-stamps including that for the ego-car which carries the ego-camera atop. As mentioned above, a node 'A' has a parameterization set that majorly includes a pose called the *estimate* for a node T_A^W $SE(3)$ strictly defined with respect to a general static global frame of reference 'W'. In our formulation, the global coordinate system happens to be centered and aligned with the ego-camera's coordinate system in the very first recorded time-stamp. The parameterization for a node also includes an ID to differentiate it from all other nodes in the same pose-graph formulation.

- **Binary-Edge:** This is a relationship defined between any two nodes in the pose-graph formulation. While a node only represents an entity in the pose-graph formulation and does not introduce a new constraint of any form into the optimization problem, an edge of any type contributes directly to the problem in the form of a unique constraint defined on one or more of the nodes. A binary edge defines a constraint specifically between two nodes say 'A' nd 'B' in a directed manner, say from 'A' to 'B'. The parameterization of the binary-edge in the above scenario would mainly contain a pose called a *measurement* for an edge T_B^A $SE(3)$ defined for node 'B' with respect to the local coordinate system for node 'A'. Furthermore, the parameterization also includes a from ID and to ID that specifies from which node to which node is the binary edge defined.

- **Unary-Edge:** This relates a node in the 3D world with a stationary feature in the environment as defined in our pose-graph formulation. This type of edge only constrains one unique node in the pose-graph formulation. The parameterization for this constraint includes a pose called a *measurement* for an edge $\mathbf{tr}_p^W$ $\mathbb{R}^2$ defined for a node to a point 'p' in the environment. This type of edge attempts to move the node's geographical location with respect to the world closer to the same of the point 'p' in the environment.

The parameterization for each constraint or edge element for any pose-graph formulation contains another important part called the *information matrix*. This is a positive semi-definite inverse-covariance matrix which defines the weight of various degrees of freedom for the pose defined for that specific type of edge. Thus, the *information matrix* is always a square matrix with size equal to the degree of freedom for the edge's pose parameter. Since a binary edge's *measurement* is in $SE(3)$, the size of it's *information matrix* is $6X6$ whereas that for a unary-edge is $2X2$ since its *measurement* is defined in $\mathbb{R}^2$. The mathematical descriptions of the respective cost functions can be found in Sec.3.3.1 and Sec.4.4.

We make use of this information matrix as a means to weigh the effect of various unary and binary edges in our pose-graph formulations. An edge with highly weighed information matrix would tend to cause more variations in the pose-graph initializations post optimization while taking lesser variations by itself. Thus, edges that take measurements from more accurate sources of initializations must have highly scaled information matrix, while those which are initialized with less reliable source of data must be scaled lower. In our work, we maintain a diagonal matrix scaled with an appropriate scalar as information matrix for various edges.

Chapter 3

Multi-Object Monocular SLAM in Dynamic Environments

We investigate the benefits of our multi pose-graph optimization framework for dynamic objects in a scene. We show that this framework enables us to cast multiple objects including ego vehicle in a *unified global frame* in *metric scale*. To the best of our knowledge, this is the first monocular multibody SLAM to represent moving obstacle trajectories in a unified global metric frame and benchmarked vis a vis ground truth. The quantitative results presented demonstrate the efficacy of the proposed formulation wherein the average trajectory error in metres for dynamic participants on KITTI sequences is well within acceptable limits considering the difficulty of the posed problem.

3.1 Overview of the Proposed Pipeline

The complete pipeline we follow can be summarized as follows:

- We take a stream of monocular images as input to our pipeline.
- We exploit 3D depth estimation to ground plane points as a source of vehicle localizations in ego-camera frame as explained in Sec. 3.2.1.
- Alternatively, as explained in Sec. 3.2.3, we fit a base *shape prior* to each vehicle instance uniquely to obtain refined vehicle localizations in ego-camera frame.
- To obtain accurate odometry estimations, we scale ORB-SLAM2[22] to metric units by estimating ego-motion with respect to detected road plane point correspondences over multiple frames in each sequence. Sec. 3.2.2 describes this module of our pipeline mathematically.
- Finally, our optimization formulation Sec. 3.3 uses the above localizations and odometry estimation to resolve *cyclic-consistencies* in the pose-graph.
- This provides us with accurate **multi-body localizations** in a **static global frame** and consistent **metric scale**.

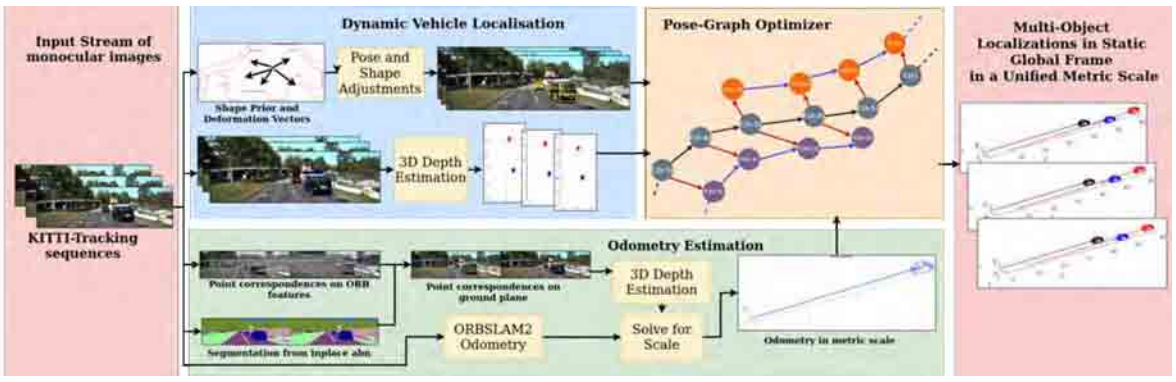

Input Stream of monocular images
KITTI-Tracking sequences
Dynamic Vehicle Localisation
Shape Prior and Deformation Vectors
Pose and Shape Adjustments
3D Depth Estimation
Pose-Graph Optimizer
Multi-Object Localizations in Static Global Frame in a Unified Metric Scale
Odometry Estimation
Point correspondences on ORB features
Point correspondences on ground plane
Segmentation from inplace abn
ORBSLAM2 Odometry
3D Depth Estimation
Solve for Scale
Odometry in metric scale

Figure 3.1: **Pipeline**: We obtain dynamic-vehicle localizations via the modules explained in blue section. The mathematical representations to the same can be found in Sec.3.2.3 and Sec.3.2.1. The green section illustrates our approach to obtain accurate odometry estimations in metric scale, as explained in Sec.3.2.2. The orange section illustrates a part of the pose-graph structure where the gray, the orange and the purple nodes represent the nodes for ego-car and two dynamic vehicles in the scene respectively. Moreover, the black, the blue and the red edges represent the camera-camera, vehicle-vehicle and camera-vehicle edges respectively.

3.2 Vehicle Localization and Odometry Estimation

3.2.1 Depth Estimation for Points on Ground Plane

We obtain 3D depth X_p^c $\mathbb{R}^3$ to a point on the ground plane 'p'[1] in camera-frame 'c' in metric scale by exploiting known camera-height h $\mathbb{R}$ in metric scale as shown in [43]. Given its 2D homogeneous coordinates in image-space as x_p^c $\mathbb{R}^3$ and camera intrinsic parameters as K $\mathbb{R}^{3\times3}$ and normal vector from ground plane as $\overline{n}$ $\mathbb{R}^3$, We obtain this can be obtained as:

$$X_p^c = \frac{-hK^{-1} \times x_p^c}{\overline{n}^T K^{-1} \times x_p^c} \tag{3.1}$$

Localization estimations from Eqn.3.1 are used as initialization to the camera-vehicle and vehicle-vehicle binary-edges in our pose-graph formulations in Sec.3.3.1.

3.2.2 Odometry Estimations

3.2.2.1 Scale Ambiguous Odometry

The initialization to our odometry pipeline (*cf.* Fig. 3.1) come from the ORB trajectory[22] in a static-global frame but in an ambiguous ORB scale as opposed to our requirement of *metric scale*. We scale the ego-motion from the ORB-SLAM2[22] input by minimizing the re-projection error of

[1]*Flat-earth assumption*: For the scope of this paper, we assume that the autonomous vehicle is operating within a bounded geographic area of the size of a typical city, and that all roads in consideration are *somewhat* planar, i.e., no steep/graded roads on mountains. Consequently, we take normal vector $n = [0, -1, 0]$.

the ground point correspondences between each pair of consecutive frames. Given frames $t-1$ and t, we have odometry initialization in 3D space in ambiguous ORB scale from ORB-SLAM2 as T_{t-1} and T_t respectively both of which are with respect to a static global frame which is the initial frame of the sequence captured by the ego-camera. We obtain the relative odometry between the two frames as follows[2]:

$$T_t^{t-1} = (T_{t-1})^{-1} \times T_t \tag{3.2}$$

3.2.2.2 Scale Initialization

We now obtain ORB features to match point correspondences between the frames '$t-1$' and 't' and use state-of-the-art semantic segmentation network[40] to retrieve sets of points $\mathbb{S}_{x(t-1)}$ and $\mathbb{S}_{x(t)}$ that lie on the ground plane at frames '$t-1$' and 't' respectively. We obtain corresponding points X_{t-1} $\mathbb{R}^3$ and X_t $\mathbb{R}^3$ in 3D, given the camera height, via Eqn.3.1 as explained in Sec.3.2.1. To reduce noise incorporated by the above method, we only consider points within a threshold in depth of T = 12 metres from the camera. Further, we obtain the required scale-factor α that scales odometry from Eqn.3.2 via a minimization problem as shown in Eqn.3.4, the objective function to which is elaborated as Eqn.3.3:

$$F(\alpha) = (X_{t-1} - (R_t^{t-1} \times X_t + \alpha \mathbf{tr}_t^{t-1})) \tag{3.3}$$

$$\min_{\alpha} F(\alpha)^T \times F(\alpha) \tag{3.4}$$

Here, R_t^{t-1} $SO(3)$ and $\mathbf{tr}_t^{t-1}$ $\mathbb{R}^3$ represent the relative rotation matrix and translation vector respectively. After solving the above minimization problem, we finalize our scale factor α $\mathbb{Z}+$ as the mean of solutions obtained from the following:

$$\alpha = \frac{(X_{t-1} - (R_t^{t-1} \times X_t))^T \times \mathbf{tr}_t^{t-1}}{(\mathbf{tr}_t^{t-1})^T \times \mathbf{tr}_t^{t-1}} \tag{3.5}$$

3.2.3 Pose and Shape Adjustments Pipeline

We make use of base *shape prior* represented by *meanshape* $\overline{X}_m$ $\mathbb{R}^{3k}$ consisting of k key-points that represent distinguishable features on vehicle instances in camera frame and B *basis vectors* $\overline{V}_m$ $\mathbb{R}^{3k \times B}$ in 3D scene to make efficient estimation of vehicle pose in camera frame. We achieve this by minimizing the pose parameters ($\hat{R}$ $\mathbb{R}^{3k \times 3k}, \hat{tr}$ $\mathbb{R}^{3k}$) and shape parameter (Λ $\mathbb{R}^B$) alternatively over a common objective function $\mathcal{L}_r$ defined based on reprojection error estimated position and observed position of each vehicle feature in 2D image space, as shown by [23, 24, 2]. This objective function can be defined as:

[2]We use $\times$ to denote matrix multiplication for the scope of this paper.

$$\min_{\hat{R},\hat{t},\Lambda} \mathcal{L}_r = \left\| \pi_K(\hat{R}(\overline{X}_m + \overline{V}_m \Lambda) + \hat{tr}; f_x, f_y, c_x, c_y) - \overline{x} \right\|_2^2 \tag{3.6}$$

The function π_K projects 3D coordinates onto 2D image space using camera intrinsic parameters $\mu = (f_x, f_y, c_x, c_y)$. $\overline{x}$ $\mathbb{R}^{2k}$ represents keypoint based observations in 2D image space as obtained from *hourglass* based keypoint localization network [26] for each vehicle instance. The pose-shape optimizer effectively computes the vehicle's pose with respect to the local camera frame by solving a modified form of camera resection problem through Eqn.3.6 [24]. Thus they automatically initialize the Camera-Vehicle edges $T^{c(t)}_{v(t)}$ and $T^{c(t-1)}_{v(t-1)}$ in the pose-graph optimizer Eqn. 3.11. The mathematical representation of the base shape prior and and more details about this procedure has been detailed in a better manner in Chapter.2.

3.3 Multi-Object Pose Graph Optimizer

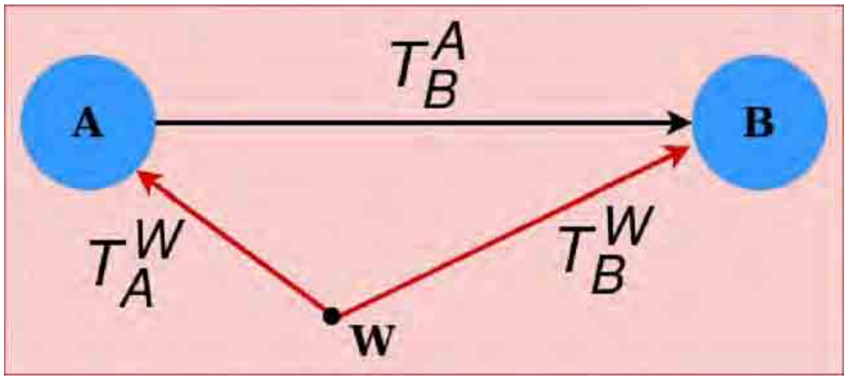


Figure 3.2: Illustration of a simple pose-graph defined by a constraint defined from nodes A to B by a binary edge.

3.3.1 Pose-Graph Formulation

Fig.3.2 illustrates the pose graph structure containing two nodes A and B and an edge between them. Using the terminologies from g2o[12], each node A in the pose graph is characterized by a pose T^W_A $SE(3)$ called the *estimate* which defines the pose of node A with respect to a static-global frame of reference W. Meanwhile, a binary-edge from A to B is represented with a relative pose T^A_B $SE(3)$ called the *measurement* which defines the pose of node A from another node B's perspective. Fig.3.3 illustrates the pose graph structure between every consecutive set of frames $t-1$ and t containing four nodes and four edges between them. We obtain the *estimates* for camera nodes (i.e., $T^W_{c(t-1)}$ and $T^W_{c(t)}$) and *measurement* for the camera-camera edge (i.e. $T^{c(t-1)}_{c(t)}$) from our odometry estimation (*cf.*3.2.2). We use this odometry to register dynamic object localizations from pose-shape adjustment pipeline as explained in Sec.3.2.3 as introduced by [2, 23, 24] to provide for the vehicle node *estimates* $T^{W}_{v(t-1)}$ and $T^{W}_{v(t)}$. We obtain *measurement* for the camera-vehicle edge (i.e., $T^{v(t-1)}_{c(t-1)}$, $T^{v(t)}_{c(t)}$) from shape and pose adjustment (*cf.* 3.2.3). Moreover, we use depth estimation on ground plane using Song et al.[43] as explained in Sec.3.2.1 as a source of vehicle localizations that is unique from the localizations obtained from Sec.3.2.3. This registered with our odometry estimations provides for our vehicle-vehicle edge *measurement* i.e., $T^{v(t-1)}_{v(t)}$.

3.3.2 Cost Functions

As explained in 3.3.1, given that W represents the static world coordinate frame of reference, nodes A and B are parameterized by poses T^W_A $SE(3)$ and T^W_B $SE(3)$ respectively, and a binary edge from node A to node B is parameterized using a pose T^A_B $SE(3)$. Each binary edge between any pair of nodes represents a unique constraint between the respective nodes. This constraint can be mathematically represented as follows:

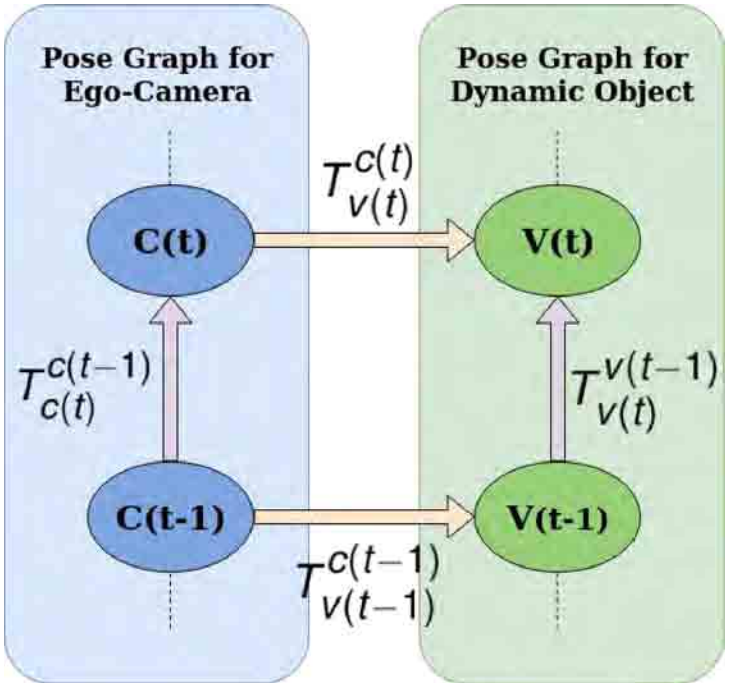


Figure 3.3: An illustration of the multi-body pose graph structure under our setting between a pair of consecutive frames. Nodes in blue correspond to the primary pose graph i.e., pose graph for the ego-motion. Nodes in green correspond to the secondary pose graph i.e., pose graph for the dynamic objects in the scene.

$$\Upsilon_{AB} = (T^A_B)^{-1} \times (T^W_A)^{-1} \times T^W_B \tag{3.7}$$

Assuming relative correctness between each term in Eqn. 3.7, it results in an identity matrix I_4 $SE(3)$ irrepective of the order of transformation. Thus, Eqn. 3.7 reduces to:

$$T_A^B \times T_W^A \times T_B^W = I_4 \tag{3.8}$$

We can see that the order in which the transformations are applied do not change the consistency of the respective cycle in the pose-graph. Thus, Eqn. 3.8 can also be written as follows:

$$T_B^W \times T_A^B \times T_W^A = I_4 \tag{3.9}$$

Using Fig. 3.3 as the illustrative reference, let four nodes be $c(t-1)$, $c(t)$, $v(t-1)$, $v(t)$ which represents ego-camera at time instant $t-1$ and t and dynamic vehicle in the scene at time instant $t-1$ and t respectively. The node *estimates* for the four nodes are $T^W_{c(t-1)}$, $T^W_{c(t)}$, $T^W_{v(t-1)}$ and $T^W_{v(t)}$ respectively.

Now, following the format of Eqn. 3.8, the cost function for each of the four binary-edges can be shown mathematically as:

$$\begin{aligned}
\Upsilon_{cc} &= T^{c(t)}_{c(t-1)} \times T^{c(t-1)}_{W} \times T^{W}_{c(t)} \\
\Upsilon_{cv(t-1)} &= T^{v(t-1)}_{c(t-1)} \times T^{c(t-1)}_{W} \times T^{W}_{v(t-1)} \\
\Upsilon_{cv(t)} &= T^{v(t)}_{c(t)} \times T^{c(t)}_{W} \times T^{W}_{v(t)} \\
\Upsilon_{vv} &= T^{v(t)}_{v(t-1)} \times T^{v(t-1)}_{W} \times T^{W}_{v(t)}
\end{aligned} \tag{3.10}$$

Here, Υ_{cc}, $\Upsilon_{cv(t-1)}$, $\Upsilon_{cv(t)}$ and Υ_{vv} represent the cost functions for camera-camera, camera-vehicle at time instants $t-1$ and t, and vehicle-vehicle edges respectively.

Cumulatively, the above cost functions for a single loop illustrated in Fig. 3.3 can be represented as:

$$\Upsilon = \Upsilon_{cc} \times \Upsilon_{cv(t)} \times (\Upsilon_{vv})^{-1} \times (\Upsilon_{cv(t-1)})^{-1} \tag{3.11}$$

On substituting Eqn. 3.10 in Eqn. 3.11, and further simplification, we obtain the resultant function for cumulative cost as follows:

$$\Upsilon = T^{c(t-1)}_{c(t)} \times T^{c(t)}_{v(t)} \times T^{v(t)}_{v(t-1)} \times T^{v(t-1)}_{c(t-1)} = I_4 \tag{3.12}$$

Clearly, Eqn. 3.12 defines the cyclic consistency within the loop defined by the four binary edges.

3.3.3 Con dence Parameterization

In addition to the relative pose between the participating graph nodes, the parameterization for each edge also includes a positive semi-definite inverse covariance matrix or the information matrix Ω_E $\mathbb{R}^{N\times N}$ where E represents an edge in the pose graph and N represents the dimension of the Lie group in which the poses are defined. In this work, all poses and transformations are defined in SE(3), hence we can take N = 6 for the information matrix Ω_E corresponding to each edge E in the whole pose graph.

We utilize the information matrix as a confidence parameterization for various sources of input-data for the pose-graph. To make the most out of this scenario, we scale the information matrix for an edge E by a scale factor $\lambda \quad \mathbb{R}$ such that effective information matrix $\overline{\Omega}_E$ that is finally passed as a parameter is obtained as:

$$\overline{\Omega}_E = \lambda \Omega_E \tag{3.13}$$

Where Ω_E corresponds to the information matrix initialization for the edge E. We categorize all the edges in our pose-graph formulation into three types namely camera-camera, camera-vehicle, and vehicle-vehicle edges. Each type of edges corresponds to a unique source of data to provide for the corresponding constraint. This formulation coupled with the corresponding *confidence parameter* λ, enables us to scale the effects of the respective categories of edges appropriately. Given that odometry estimates are fairly reliable, we assign a relatively high constant scaling to its information matrix for our experiments on all sequences.

Given that we obtain dynamic vehicle localizations in camera frame from two different sources for camera-vehicle edges and vehicle-vehicle edges respectively as explained in 3.3.1, we make use of this parameter to scale the information matrix corresponding to the two types of edges in each loop in our pose graph. It has been observed over a large number of vehicles in the data that localizations obtained from 3.2.3 performs better than the localizations obtained from 3.2.1 for vehicles that are at a relatively closer depth to the camera (up to about 45 metres). However, localizations from 3.2.1 are relatively more accurate when compared to the same obtained from 3.2.3 at depths far away from the camera (over 45 metres).

This can be attributed to the keypoint localizations being inaccurate for vehicles that are very far from the camera due to lack of clarity indistinguishable features on these vehicles. However, factors like visible features do not affect the dynamic vehicle localizations using 3.2.1 as this method relies only on the camera intrinsic parameters, ground plane normal, camera height and 2D bounding boxes[37].

3.4 Experiments and Results

3.4.1 Dataset

We test our procedure over a wide range of KITTI-Tracking training sequences[10], spanning over rural and urban scenarios with various number of dynamic objects in the scene. We perform localizations on objects primarily consisting of cars and mini-vans. Our localization pipeline provides accurate results over objects irrespective of the direction of motion and maneuvers undertaken by both the ego-car as well as the other vehicles under the ego-car's observation in a multi-object scenario.

The labels provided along with the training sequences for KITTI-Tracking dataset [10] are used as ground truth for getting depth to the vehicle's center from the camera. The corresponding ground truth for odometry comes from GPS/IMU data, which is compiled using the OXTS data provided for all the KITTI-Tracking training sequences.

3.4.2 Approaches Evaluated

We evaluate the performance of the following approaches:

- *Namdev* et al.[25]: A monocular multibody VSLAM approach that obtains motion for dynamic objects and ego-camera in a unified scale. The non tractable relative scale that exists between the moving object and camera trajectories is resolved by imposing the restriction that the object motion is locally linear.
- *Our Batch-Wise Approach with Scale Initialized Odometry*: A monocular multibody approach with a batch-wise pose-graph optimization formulation that resolves relationships with dynamic objects as a means of performing SLAM.

3.4.3 Qualitative Results

3.4.3.1 Pose and Shape Adjustments

Fig.3.6 illustrates ego-motion as well as the motion of various vehicles over many sequences from the KITTI Tracking dataset[10] after performing pose-graph optimizations in the batch-mode of experiments with scele initialized version of odometry initialization. We obtain accurate localizations in ego-camera frame by fitting base *shape priors* to each non-occluded and non-truncated vehicle in the scene irrespective of its orientation with respect to the ego-camera. While the pipeline is dependent on the keypoint localizations on these vehicles, factors like large depth from camera are bound to affect the accuracies with respect to ground truth. However, this approach ensures fairly accurate vehicle localizations for the pose-graph optimizer to apply its edge constraints on. Fig.3.4 illustrates wireframe fitting and subsequent mapping in ego-frame for a traffic scenario consisting of multiple vehicles. A more clear comparison with ground truth can be made through Fig.3.7 which illustrates how our pipeline is capable of obtaining ego-trajectory estimates along with the trajectories for multiple dynamic vehicles in the ego-camera's scene.

3.4.3.2 Odometry Estimation

For accurate visual odometry, we exploit distinguishable static ORB[21, 22] features on the road plane from entities like curbs, lane markers and any irregularities on the road to obtain quality point correspondences. While the approach is dependent on factors like reasonable visibility, we obtain robust

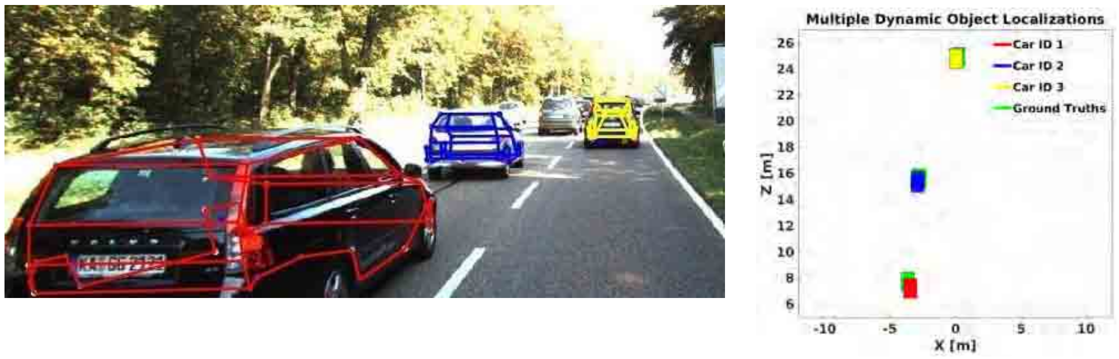


Figure 3.4: Localizations in ego-camera frame after pose and shape estimations for a dynamic multi-vehicle scenario.

performance over a diverse range of sequences many of which are over a 100 frames long. Fig.3.5 illustrates how our method achieves a fairly accurate scaling of odometry to provide an initialization that competes well with the corresponding ground truth.

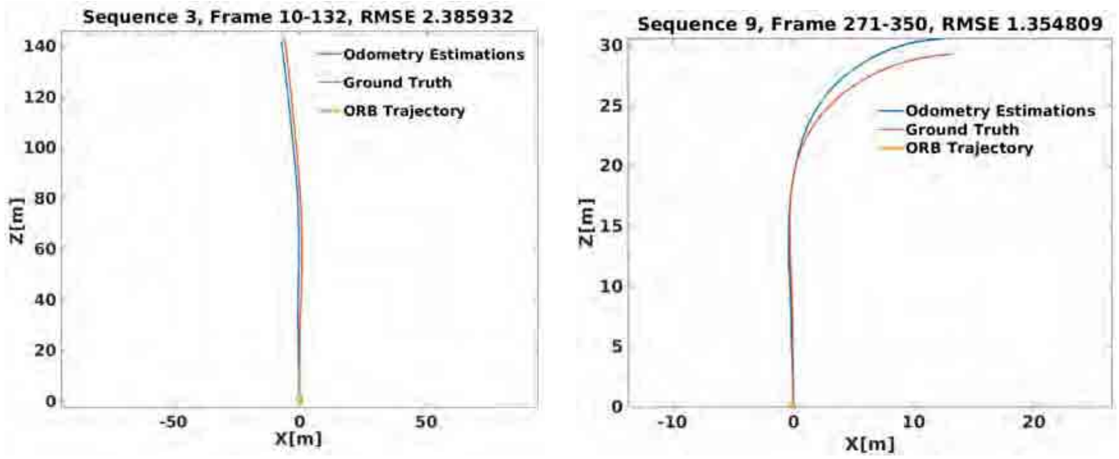


Figure 3.5: Odometry estimations in metric scale in blue. GPS/IMU trajectory is in red and ORB trajectory in its scale is in yellow. The figure illustrates that our odometry estimation is proficient on sharp turns and long sequences.

3.4.3.3 Pose-Graph Optimization

We resolve for each cyclic-loop created by the ego-camera and each vehicle (*cf.* Eqn. 3.12) in the scene in our optimization formulation. The optimization problem runs for a maximum of 100 iterations. Our unified pose-graph based optimization formulation performs consistently well on a wide range of sequences irrespective of the length of the sequences, number of objects being localized and varying instance lengths for each vehicle in the same sequence. A unique pose-graph structure for all vehicles including ego-motion at each time instance ensures effective error re-distribution across all trajectories based on efficient confidence allocation in the form of information matrix scaling (*cf.* Eqn. 3.13). Fig.3.7 and Fig.3.7 illustrates ego-motion as well as the motion of various vehicles over many sequences from the KITTI Tracking dataset[10].

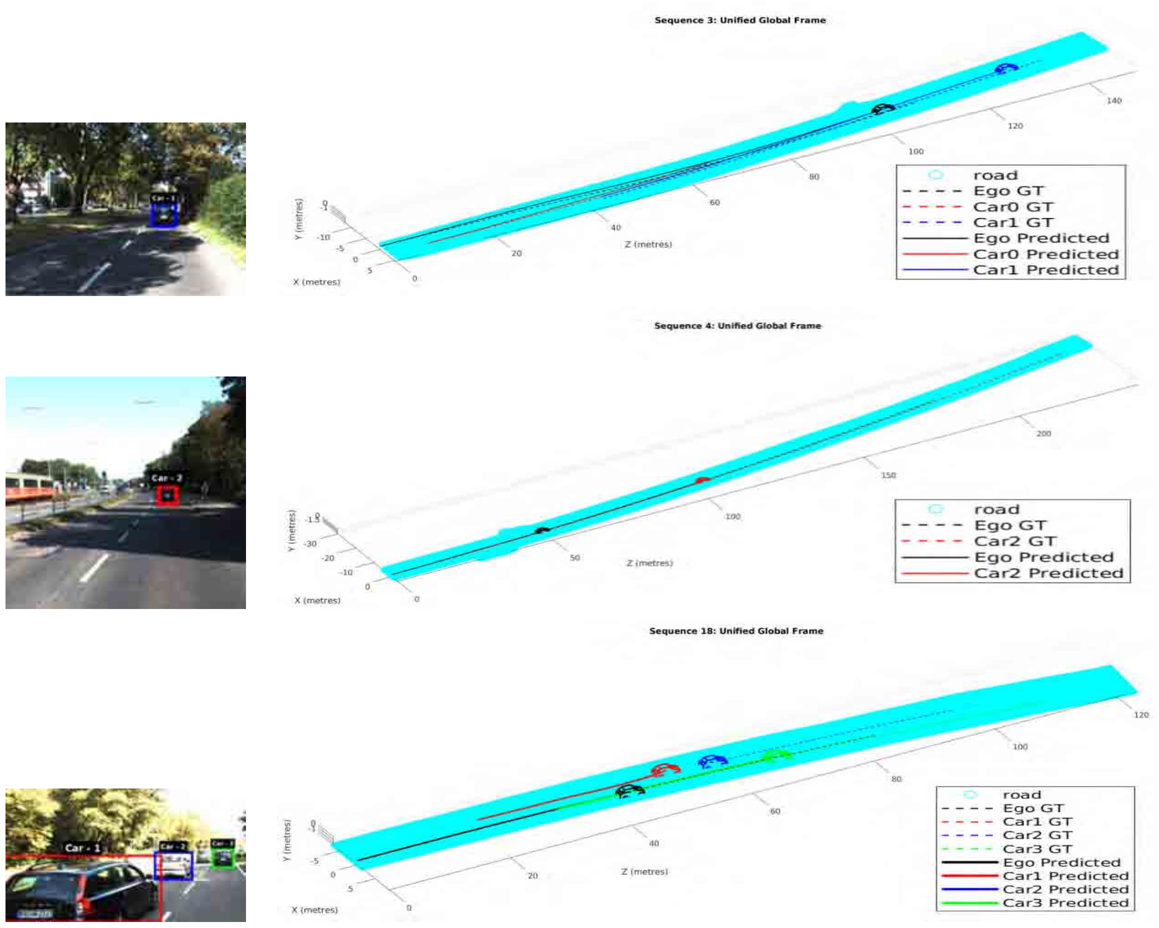
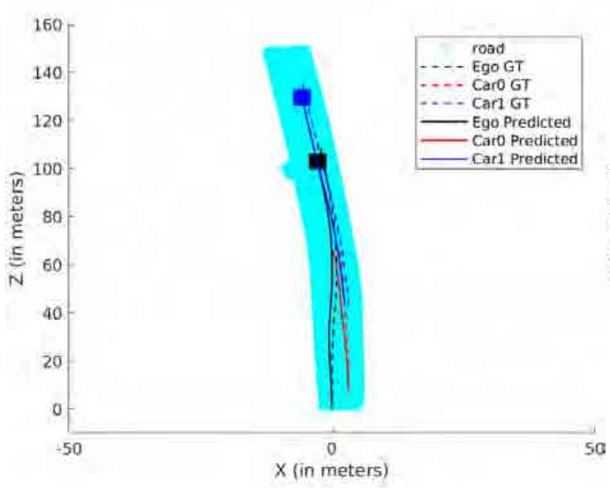
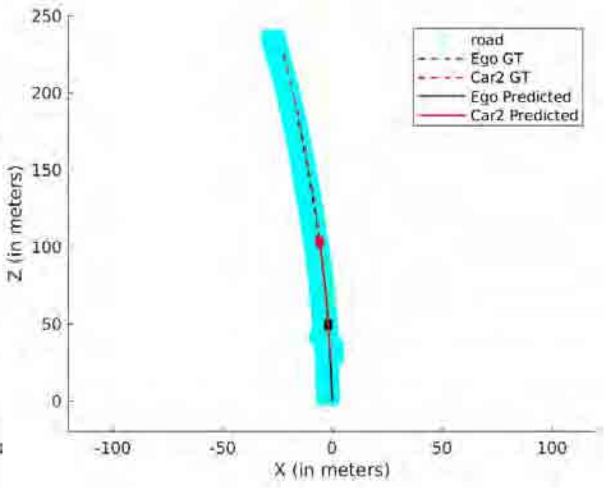


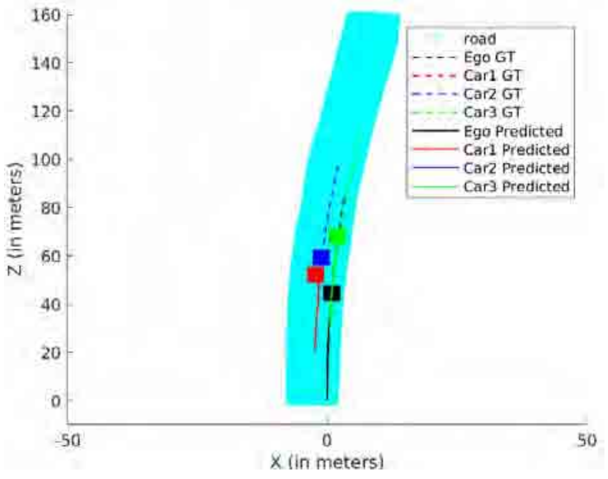
Figure 3.6: Qualitative results on various sequences for **Batch-Wise Approach with Scale Initialized Odometry**. *Col 1* shows input images with bounding boxes for vehicles mapped in *Col 2* and *Col 3*. While *Row 1* and *Row 3* illustrate our performance on multi-vehicle traffic scenarios, *Row 2* shows results for a far away vehicle over a long sequence. Ego-vehicle is shown in black whereas the red, blue and green plots show unique dynamic vehicles in the scene with corresponding dotted plots showing the ground truths. Note that entire ground truth trajectory is shown at once in the figures whereas our results are up to the instance frame shown in *Col 1*.



(a)



(b)



(c)

Figure 3.7: Qualitative top-view plots for **Batch-Wise Approach with Scale Initialized Odometry**. The three plots show the top view version for the same sequences shown in the perspective 3D mapping illustrated in Fig.3.6. The road plane is represented in cyan whereas the odometry estimates are show as a black trajectory. Other dynamic vehicles in the camera's scene are show as red, green and blue trajectories. Dotted trajectories represent the respective ground truth trajectories.

3.4.4 Quantitative Results

3.4.4.1 Odometry Estimations

To improve odometry estimations, we place a threshold T on depth from camera upto which we consider point correspondences. This is set based on our observation that the accuracy of the 3D depth to the point correspondences lowers with depth from the camera. Table 3.1 summarizes our experiments with various threshold values before we finalize our threshold at T = 12 metres.

While T = 12 meters delivers best results for most sequences mentioned in Table 3.1, we see that T = 15 metres performs better for sequences 6 and 9, both of which involve the ego-vehicle taking a sharp turn at an intersection. This is because we rely on ground plane features including and largely contributed to by the lane markers on the road plane. Given that the segment of road plane in the scene at an intersection is devoid of any road/lane markers, we do not get enough feature correspondences from closer segments of the road. Meanwhile, increasing the threshold enables us to pick up points from the road plane continuing beyond the intersection which contains better scope for quality feature correspondences in the form of lane markers. Consequently, a relatively larger threshold performs better.

Absolute Translation Error (ATE) (metres)					
Seq no.	Seq length	Threshold (metres)			
		12	15	18	20
1	41	**4.39**	5.63	5.61	5.18
3	123	**1.65**	2.45	1.91	2.57
4	149	**7.64**	8.84	9.59	10.96
6	51	5.90	**2.37**	2.38	2.82
9	80	5.52	**1.35**	1.44	1.44
18	141	**1.98**	3.31	2.98	3.36
Average ATE		4.51	3.99	**3.98**	4.39

Table 3.1: Analysis between various threshold settings for odometry estimations by computing Absolute Translation Error (ATE) in metres.

3.4.4.2 Pose-Graph Optimization

Our pose-graph formulation consists of three categories of edges namely camera-camera(C-C) edges, camera-vehicle(C-V) edges and Vehicle-Vehicle(V-V) edges. Each of these sets of edges are scaled with a unique *confidence parameter* λ. To analyse how each category of edges constraint our pose-graph and consequently affect our optimization results, we analyse the effect of removing the constraint put forth by each type of edges $E \quad C\text{-}C, C\text{-}V, V\text{-}V$ by setting the corresponding $\lambda_{E'} = 0$. Table.3.4 and Table.3.5 summarizes our experiments with the pose-graph optimizer and various combinations of its edge-parameterizations. It can be noted from Table.3.3 and Table.3.2 that few vehicles in sequence 3 and the ego-vehicle in sequence 4 perform better when C-C constraints are relaxed. This is because the odometry estimations are relatively less accurate in these sequences when compared to the other vehicles in the same sequence. The optimizer generally utilizes reliable edges in each loop of the pose-graph to improve the relatively less reliable edges, provided their information matrices are scaled appropriately. Given that the C-C edges are relatively less reliable in these sequences, relaxing its constraints from the optimization problem enables other edges to improve upon the overall error in these sequences. A similar explanation can be given for the errors shown by the ego-motion in sequence 18. Since C-V edges of the ego-motion is relatively more accurate than the corresponding C-V edges of other vehicles, we obtain a better result for the same when the C-V edge constraint is relaxed. Both C-C and V-V edges are generated using the odometry estimations and are thus in uenced by its accuracy too.

Absolute Translation Error (ATE) (Root Mean Square) in Global Frame (metres)							
Seq number	3		4	18			Avg Error
Car ID	0	1	2	1	2	3	
Frame length	41	92	149	62	83	141	
Initialization	1.62	4.99	13.65	1.33	3.47	3.53	4.76
CC , CV	1.62	5.01	13.65	1.32	3.48	3.24	4.72
CC , VV	2.88	5.22	2.14	1.29	4.00	2.80	3.06
CV , VV	1.61	5.68	2.24	1.65	**3.03**	**2.24**	2.74
CC , CV , VV	**1.61**	**4.99**	**2.14**	**1.29**	3.45	2.40	**2.65**

Table 3.2: ATE for all dynamic vehicles in the ego-camera's scene in a static-global frame recognized by the formulation across various sequences. The tables considers the ATEs obtained for the experiments when various sets of edges are taken into consideration by the optimization problem.

Absolute Translation Error (ATE) (Root Mean Square) in Global Frame (metres)				
Seq number	3	4	18	Avg Error
Frame length	123	149	141	
Initialization	1.96	6.43	2.24	3.54
CC , CV	1.98	6.43	2.24	3.55
CC , VV	1.96	6.43	**2.24**	**3.54**
CV , VV	3.54	**6.41**	2.76	4.24
CC , CV , VV	**1.96**	6.49	2.27	3.57

Table 3.3: ATE for all ego-trajectory estimates in a static-global frame recognized by the formulation across various sequences. The tables considers the ATEs obtained for the experiments when various sets of edges are taken into consideration by the optimization problem.

Table.3.4 and Table.3.5 compare our performance with Namdev*et al.*[25]. Since ATE is not reported in their literature, we calculate the ATE after running the available implementation. As is evident from Table.3.4 and Table.3.5, we showcase superior performance in all sequences when compared with Namdev *et al.*[25].

Absolute Translation Error (Root Mean Square) in Global Frame (metres)							
Seq No.	3		4	18			Avg Error
Car ID	0	1	2	1	2	3	
Frame length	41	92	149	62	83	141	
Namdev*et al.*[25]	13.81	11.58	11.18	3.77	5.93	3.72	8.33
Ours	**1.61**	4.99	2.14	1.29	3.45	2.4	2.65

Table 3.4: Quantitative performance for various dynamic vehicles in the scene of the ego-camera for all evaluated approaches as summarized in Sec.3.4.2 in a static-global frame in a single *metric scale*. The errors are computed as root-mean-squared Absolute Translation Errors averaged throughout across complete trajectories for each of the vehicles instances reported in the table.

Absolute Translation Error (Root Mean Square) in Global Frame (metres)				
Seq No.	3	4	18	Avg Error
Frame length	123	149	141	
Namdev*et al.*[25]	11.49	11.12	3.69	8.77
Ours	1.96	6.49	**2.27**	3.57

Table 3.5: Quantitative performance for ego-camera trajectories in the scene for all evaluated approaches as summarized in Sec.3.4.2 in a static-global frame in a single *metric scale*. The errors are computed as root-mean-squared Absolute Translation Errors averaged throughout across complete trajectories for each of the vehicles instances reported in the table.

3.4.5 Summary of Results

While Fig.3.6 and Fig.3.7 illustrate how our trajectories perform with respect to ground truth, Table.3.4 and Table.3.5 reaffirms how our pose-graph formulation successfully redistributes errors about constraints with high *confidence parameters*. Table.3.4 and Table.3.5 vindicate the efficacy of the proposed pipeline as the absolute translation error(ATE) are typically around 3m for sequences more than 100m in length. The last row of the table denotes the percentage error, which is significantly low for fairly long sequences at an average of 3.11%, considering that the original problem is intractable and hard to solve.

Chapter 4

Mapping Trajectories of On-Road Dynamic Participants with a Moving Monocular Camera: A Multibody Approach

Given a temporal sequence of on-road moving camera observations, we estimate ego-motion as well as motion of other dynamic participants in the scene. In it, ego-car estimations are represented by black coloured wireframe while red, blue and green wireframes represent other dynamic objects in the scene. Moreover, different lanes in the scene are represented by red, green and purple planes. The keynote here is that the vehicle trajectory's scale is typically not in the same scale as that of the camera trajectory. To represent both these trajectories in the same scale with respect to a global reference frame forms the cornerstone of this effort.

4.1 Overview of the Proposed Pipeline

4.1.1 Vehicle Localization

Given an input sequence of images, the pipeline in Fig.4.1 obtains appropriate object localization from a 3D depth estimator which is detailed in Sec.4.2.1. This module obtains 3D depth initialization to the ground plane points given the coordinates in 2D image space to a point nearest to the vehicle. We also obtain reliable vehicle localization in camera frame from the *Pose Shape Optimizer* module as explained in Sec.4.2.2 similar to [24, 23, 2] which fits a base *shape prior* to any object instance in the scene. These object localization provide for the camera-object edge *measurements* as explained in Sec.4.4.

4.1.2 Visual Odometry

Our pipeline follows Monocular ORB-SLAM2 [22] based initialization for ego-camera trajectory estimates as detailed in Sec.3.2.2.1. However these estimates are in a different scale space vis a vis that in which the objects are represented after Pose Shape Optimizer mentioned above. We make use of a *scale initializer* sub-module as explained in Sec.3.2.2.2 which takes stationary features on the ground

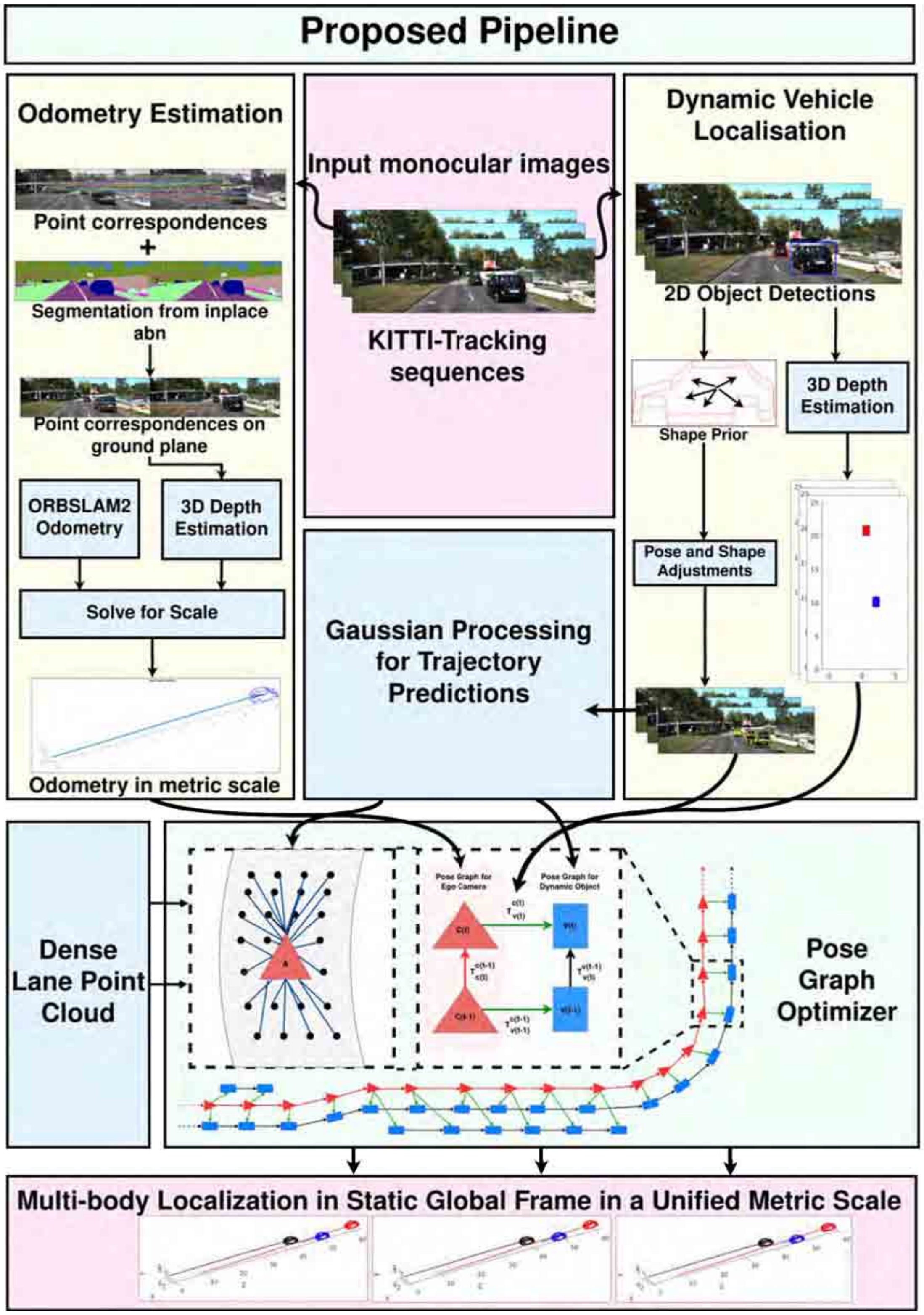

Proposed Pipeline
Odometry Estimation
Point correspondences
+
Segmentation from inplace abn
Point correspondences on ground plane
ORBSLAM2 Odometry
3D Depth Estimation
Solve for Scale
Odometry in metric scale
Input monocular images
KITTI-Tracking sequences
Gaussian Processing for Trajectory Predictions
Dynamic Vehicle Localisation
2D Object Detections
Shape Prior
3D Depth Estimation
Pose and Shape Adjustments
Dense Lane Point Cloud
Pose Graph for Ego Camera
Pose Graph for Dynamic Object
Pose Graph Optimizer
Multi-body Localization in Static Global Frame in a Unified Metric Scale

Figure 4.1: Detailed Illustration of the proposed pipeline.

plane as cues and exploits known parameters in metric scale to obtain accurate odometry estimates in metric scale. Such scale aligned localization of ego-camera and other dynamic objects are used by the pose-graph optimizer to improve object and camera trajectories.

4.1.3 Trajectory Prediction

We also introduce a *trajectory predictor* module based on Gaussian Process based regressor as shown in Sec.4.3 that predicts the pose of dynamic objects for the subsequent time instant $n+1$ given the corresponding object poses for past n time instances. The vehicle pose computed by the Pose Shape estimator module is mediated to the global reference frame using scale-initialized ego-camera estimation.

4.1.4 Pose-Graph Optimizer

The quintessential *pose-graph optimizer* module of the pipeline makes use of factor graph formulation as shown in Sec.3.3.2 to optimize for ego-camera and object poses. It makes use of both dynamic features in the form of vehicle poses and static features in the form of road-plane points that fall between the consecutive lane boundaries. In its basic form, it optimizes over the *cyclic consistency* constraint based on the current and previous ego-camera and object poses. It also makes use of static lane-based cues from the environment to correct its pose estimates with respect to the static features. Two versions of this optimizer are used in this paper. In the first version, the scale initializer is only used as a bootstrap for vehicle poses while the rest of the vehicle pose estimation is initialized with *trajectory predictor* module. In the second version, our pipeline uses scale initialized through the *scale initializer* module as the transformation to the pose graph nodes for the entire length of dynamic object traversal. However, the camera poses are not scale initialized or scale aligned but represented in ORB-SLAM2's scale space itself. Thus, the optimizer lifts the ego trajectory to the same metric scale in which vehicle trajectories are represented apart from optimizing for the ego and vehicle poses.

4.2 Obtaining Trajectory Initializations in Global Frame

4.2.1 3D Depth Estimator

Given the height at which the monocular camera setup is mounted in *metric scale* we try to make an intelligent estimation of 3D location of any point lying on he ground plane[43] given that it is perfectly horizontal[1]. Given this height as h $\mathbb{R}$ metres, we obtain an initialization to the 3D location X_p^c $\mathbb{R}^3$ of a point x_p^c $\mathbb{R}^3$ on 2D image-space, provided parameters such as the camera intrinsics K $\mathbb{R}^{3\times 3}$ as well as the normal to the ground plane $\overline{n}$ $\mathbb{R}^3$ are also known. Given our assumption(*cf.*1), we take

[1]*Flat-earth assumption*: For the scope of this paper, we assume that the autonomous vehicle is operating within a bounded geographic area of the size of a typical city, and that all roads in consideration are *somewhat* planar, i.e., no steep/graded roads on mountains. Consequently, we take normal vector $n = [0, -1, 0]$.

normal vector $n = [0, -1, 0]$. Note that we obtain all our observations for any point 'p' on the round plane in the coordinate system conforming to the camera 'c'.

$$X_p^c = \frac{-hK^{-1} \times x_p^c}{\overline{n}^T K^{-1} \times x_p^c} \tag{4.1}$$

These 3D coordinates, obtained in camera frame can initialize the Pose-Shape Optimizer below when they get depth to the base of the 2D bounding box vehicle detector in the image along the lines of [24]. They can also be used to initialize the translation component of the Camera-Vehicle edges $T_{v(t)}^{c(t)}$ and $T_{v(t-1)}^{c(t-1)}$ in Eqn.4.10.

4.2.2 3D Object Localization from Pose-Shape Optimizer

In this module, we obtain accurate vehicle localization in the coordinate frame of the monocular camera in *metric scale* by fitting a base *shape prior*[24, 23, 2] onto each vehicle instance on the camera's scene. This *shape prior* consists of a base wireframe model that captures the general structure of the car and a set of vectors that deform this generalized *shape prior* to fit to the respective vehicle instance in the scene. We make use of the depth estimation approach brought up in Sc.4.2.1 to initialize the pose for each vehicle instance in the scene and further perform a set of non-linear least squares optimization based on ceres-solver[1] to optimize for *pose parameters* and *shape parameter* of the wireframe.

$$\min_{\hat{R},\hat{t},\Lambda} \mathcal{L}_r = \left\| \pi_K(\hat{R}(\overline{X}_m + \overline{V}_m \Lambda) + \hat{tr}; f_x, f_y, c_x, c_y) - \overline{x} \right\|_2^2 \tag{4.2}$$

As explained in [24], given the base *shape prior* in the form of a mean wireframe $\overline{X}$ $\mathbb{R}^{3k}$ and B $\mathbb{Z} - *$ deformation vectors V $\mathbb{R}^{3k \times B}$ for k $\mathbb{Z} - *$ ordered keypoints, we try to optimize for the *pose parameters* R $SO(3)$, $\mathbf{tr}$ $\mathbb{R}^3$ and the *shape parameter* Λ $\mathbb{R}^B$ using the cost function defined mathematically in Eqn.4.2. We obtain the observations in the form of an ordered set of keypoints $\hat{x}$ $\mathbb{R}^{2k}$ in 2D image-space as obtained from a keypoint localization network[26]. ore detailed explanations on the *shape priors* can be found in Chapter.2.

4.2.3 Ego Trajectory Estimation and Scale Initializer

We obtain odometry initializations in global frame from ORB-SLAM2 [22] that are in an ambiguous non-metric scale. We exploit 3D depth $X_g^{c(t)}$ $\mathbb{R}^3$ in camera frame c at each time instant t from Sec. 4.2.1 to static feature correspondences g on the ground plane in the form of a cost function to reduce the reprojection error between consecutive frames $t-1$ and t. We finally solve for the scale factor α_g using the following solution as explained in Sec.3.2.2.2:

$$\alpha_g = \frac{(X_g^{c(t-1)} - (R_{c(t)}^{c(t-1)} X_g^{c(t)}))^T tr_{c(t)}^{c(t-1)}}{(tr_{c(t)}^{c(t-1)})^T tr_{c(t)}^{c(t-1)}} \tag{4.3}$$

Here, relative rotation matrix and translation vector between frames $t-1$ and t are represented by $R_{c(t)}^{c(t-1)}$ and $tr_{c(t)}^{c(t-1)}$. The final scale factor obtained as the mean of individual α_g solutions for each ground plane feature g, is used to scale or lift the camera trajectory to the metric scale in which the vehicle poses are represented. These poses are then used to obtain the Camera-Camera transformations $T_{c(t)}^{c(t-1)}$ between successive instances in the pose-graph Eqn. 4.10. These poses also register vehicle localizations from Sec. 4.2.2 into global frame which goes into the trajectory predictor module in Sec. 4.3.

4.3 Gaussian Processing(GP) based Vehicle Trajectory Prediction

As an attempt to formulate a pose-graph based trajectory prediction formulation, we explored a few approaches that could aid us for the same. We finally went ahead the the Gaussian Process(GP) based approach for the same.

Given a sequence of last k vehicle states as updated by the pose-graph formulation, we predict the next vehicle state as:

$$\begin{aligned} \hat{V}_{n+1} =& f(V_n, ..., V_{n-k+1}) \\ V_n =& (x_{c(n)}, y_{c(n)}, z_{c(n)})^T \end{aligned} \tag{4.4}$$

We obtain trajectory prediction in X-Z plane. While we maintain a constant object height[2] throughout our experiments, we obtain object orientation information from Sec.2.1.1. We model the regression problem as time-series prediction (TSP) along the lines of [39]. Unlike static model fitting methods that maps observation at time instances n to a static map $f(n)$, GP based TSP method models a distribution over $f(n)$ and infers the most likely distribution to give observations at time instances $y(n)$. The distribution thus adapts itself as recent observations become available and then adapt itself to the changes in vehicle behaviour with time. In short, GP lends itself as a very competent framework to model both stationary and dynamic time-series data. As far as implementation goes vehicle observations over the last n instances $x(n), z(n) \quad \mathcal{GP}(\boldsymbol{\mu}(\boldsymbol{n}), \mathbf{K}(\mathbf{n}, \mathbf{n}))$ where $\boldsymbol{\mu}$ is the mean function and $\mathbf{K}(\mathbf{n}, \mathbf{n})$ is the covariance matrix whose elements are the Kernel function $K(n_i, n_j)$,where $i, j \quad 0, ..., n$. Then given the next instant as $n+1$, the prior distribution for x, z is given as:

$$\hat{x}_{n+1} \quad N\left(\begin{bmatrix} \boldsymbol{\mu}(\boldsymbol{n}) \\ \mu(n+1) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{n}, \boldsymbol{n}) & \boldsymbol{K}(\boldsymbol{n}, n+1) \\ \boldsymbol{K}(n+1, \boldsymbol{n}) & k(n+1, n+1) \end{bmatrix}\right) \tag{4.5}$$

While the predictive posterior distribution for $x(n+1), z(n+1)$ belongs to $\mathcal{GP}N(m^*, \sigma^*)$ where:

$$\begin{aligned} m^* =& \mu(n+1) + \boldsymbol{K}(n+1, \boldsymbol{n})\boldsymbol{K}(\boldsymbol{n}, \boldsymbol{n})^{-1}[y - \boldsymbol{\mu}] \\ \sigma^* =& \hat{K}(n+1, n+1) - \boldsymbol{K}(n+1, \boldsymbol{n})\boldsymbol{K}(\boldsymbol{n}, \boldsymbol{n})^{-1}\boldsymbol{K}(\boldsymbol{n}, n+1) \end{aligned} \tag{4.6}$$

[2]*Flat-earth assumption*: See footnote 1

Similarly, the predictive posterior distribution for z at instant $n + 1$ can be obtained. The GP module [35] predicts vehicles at next state $\hat{V}_{n+1}$ from the above distribution, which is used by the pose-graph optimizer to optimize for the vehicle pose V_{n+1}, which is fed back to the GP regression module to predict for the subsequent instance $n + 2$. Default settings for the mean function and the kernel functions of the K matrix [35] are used to regress to the next state of the vehicle motion. The next vehicle state predicted by the GP regresser along with the pose graph optimized current vehicle pose is used to obtain the Vehicle-Vehicle relations $T_{v(t-1)}^{v(t)}$ in the pose-graph optimizer Eqn. 4.10.

We make use of the GPML MATLAB Toolbox[34] to portray the required time series prediction for dynamic vehicle trajectories as regression problems. For the purpose of our experiments, we have stick to a constant $n = 5$ frames policy to make a one-step ahead prediction each time based on the $n = 5$ step data fed as input. As mentioned above, the module functions in a 2 dimensional X-Z space where predictions are made along X and Z axes separately in a manner independent of each other. Our hyperparameter initializations include an empty mean function as well as a Gaussian likelihood function which are common to both predictions in X and Z. However, these initializations are made seperately for prediction models for bot X and Z componants of the trajectory. While the X componant of GPML predictions are initialized with Squared Exponential Covariance function, that for the X componant ar einitialized with Fractional Brownian motion covariance function. We also resort to a complete feedback of predicted samples of our trajectory in our real-time mode of experimentations where we make use of the actual input data only to bootstrap the initial $n = 5$ frames of the whole trajectory which is more than 100 frames long. The rest of the trajectory functions on the predictions made in the previous iteration of Gaussian processing.

However, such a nature of prediction formulation can potentially lead to an exponentially increasing error model which renders the estimated trajectory unrelatable with respect to the expected trajectory after a few frames of predictions. We thus make sure each prediction is subject to at least one iteration of our pose-graph optimization such that, its errors remain within the reach of our pipeline's optimization module while at the same time remaining in sync with the relative scale followed by the other objects considered by our pose graph formulation which is explained in a detailed manner in Sec.4.4.

Alternative approaches such as the following were experimented with for this module before finalizing GP based prediction model:

- EKF based trajectory prediction: An EKF based prediction approach was experimented with written in python. While this approach performed well for trajectory samples following a fairly uniform trend in motion, it tended to fail quite consistently on trajectories that followed curved paths and that showed reasonably frequent changes in velocity.

- *LSTM based prediction model*: An LSTM Cell and an LSTM based model were experimented with to predict trajectory samples. It was observed that these models could make very accurate predictions for trajectory points captured in local camera frame. Unfortunately, the same models did not perform so well for trajectory points captured in a static global frame which was the requirement.

The above observations compelled us to stick with the GP based prediction model, which gave fairly good predictions for reasonable variations in velocity in both magnitude and direction and at the same provided us fairly accurate predictions in static global frame for long trajectories. It must however be noted that an LSTM based approach which captures object velocities is expected to perform with a much better accuracy than all approaches listed above. This approach would potentially correct for the lateral drift if trained properly. This was realized later in the project timeline and thus was not explored with sufficient emphasis. Moreover, use of a prediction model that would correct for the lateral drift by itself would potentially prevent us from experimenting with the lane based constraints explained in Sec. 4.6 with enough emphasis. Thus, it seemed optimal to go ahead with the GP based approach for prediction model.

4.4 Pose-Graph Formulation

4.4.1 Binary-Edges

Following g2o terminologies, the *estimate* $T_S^W \quad SE(3)$ characterizes pose for node S in global frame W. The *measurement* $T_D^S \quad SE(3)$ denotes a binary-edge from source node S to destination node D effectively constraining the respective *estimates*. This can be represented mathematically as the following cost function:

$$\Upsilon_{SD} = (T_D^S)^{-1} \times (T_S^W)^{-1} \times (T_D^W) \tag{4.7}$$

A more detailed explanation to this can be found in Sec.4.4

4.4.2 Unary-Edges

We also use unary-edges between agent nodes and stationary scene-landmarks denoted as 'p' with 3D depth coordinates $X_p^W \quad \mathbb{R}^3$ in global frame 'W'. Here, the agent 'A' could be ego-camera or the dynamic object in scene. This does not constrain the orientation of the agent. The resultant constraint between an agent node 'A' with translation vector $tr_A^W \quad \mathbb{R}^3$ and a world landmark 'p' in global frame can be shown as:

$$\Psi_A = tr_A^W - X_p^W \tag{4.8}$$

4.4.3 Con dence Parameterization

Our formulation also includes a positive semi-definite inverse covariance matrix or an *information matrix* in each edge's parameterization, shown as $\Omega_E \quad \mathbb{R}^{N \times N}$ where N is the dimension of the respective Lie group in which edge E is defined. Here, the pose and transformation for all binary-edges are defined in $SE(3)$ while the same for all unary-edges are defined in $(R)^3$. We can thus take $N = 6$ for information matrix Ω_E for each binary-edge and $N = 3$ for the same for all unary-edges. We exploit this to convey confidence of each constraint. We do so by scaling Ω_E upto the *effective information matrix* $\overline{\Omega}_E$ by a factor $\lambda \quad \mathbb{R}$ as:

$$\overline{\Omega}_E = \lambda \Omega_E \tag{4.9}$$

4.5 Exploiting Dynamic cues in Pose-Graph Optimizer

As shown in Fig.4.2 we categorize our pose-graph formulation into three sets of relationships represented by camera-camera, vehicle-vehicle and camera-vehicle edges. Considering consecutive time-instances '$t-1$' and 't', we aggregate individual cost functions for each edge from Eqn.4.7 to obtain a cumulative function which reduces to the mathematical representation of a simple *cyclic-consistency* for the graph-loop in the following manner:

$$\begin{aligned} \Upsilon =&(\Upsilon_{cc}) \times (\Upsilon_{cv(t)}) \times (\Upsilon_{vv})^{-1} \times (\Upsilon_{cv(t-1)})^{-1} \\ =&(T_{c(t)}^{c(t-1)}) \times (T_{v(t)}^{c(t)}) \times (T_{v(t-1)}^{v(t)}) \times (T_{c(t-1)}^{v(t-1)}) \\ =&I_4 \end{aligned} \tag{4.10}$$

We resolve this for each object in scene being considered with respect to ego-camera effectively solving a single optimization problem for all vehicles together. We categorize all edges in our pose-graph formulation into three types namely camera-camera, camera-vehicle, and vehicle-vehicle. Each type corresponds to a unique source of input data for the constraint. This formulation coupled with the *confidence parameter* λ, enables us to scale the effects of respective categories of edges appropriately.

Initialization for camera-vehicle edge parameters $T_{v(t)}^{c(t)} \quad SE(3)$ and $T_{v(t-1)}^{c(t-1)} \quad SE(3)$ in Eqn.4.10 come from Sec. 4.2.2. We obtain another set of depth estimates from Sec.4.2.1 while using orientation from Sec. 4.2.2. While Sec. 4.2.1 performs better at farther depths from camera, Sec.4.2.2 is much more superior for upto $\approx 40m$. we thus resort to a coupled confidence parameterization for camera-vehicle edges to obtain optimal estimates across a wide depth-range. The trajectory predictor from Sec.4.3 provides for the vehicle-vehicle edge parameter $T_{v(t)}^{v(t-1)} \quad SE(3)$ in Eqn.4.10.

Broadly, we experiment in two modes based on whether the camera-camera binary-edge parameter $T_{c(t)}^{c(t-1)} \quad SE(3)$ in Eqn.4.10 are fed with *scale-initialized* odometry from Sec.3.2.2.2 or *scale-ambiguous* estimations [22] from Sec.3.2.2.1. While we experiment with *scale-ambiguous* odometry only on batch-based approaches, we experiment on both batch-based and real-time approaches when it comes to *scale-initialized* odometry. In our experiments with *scale-ambiguous* odometry initialization, we were compelled to provide minimal confidence to camera-camera edges when compared to that for the other edges. While for other experiments, given that odometry initializations are fairly reliable, we assign a relatively high constant scaling to its information matrix for our experiments on all sequences.

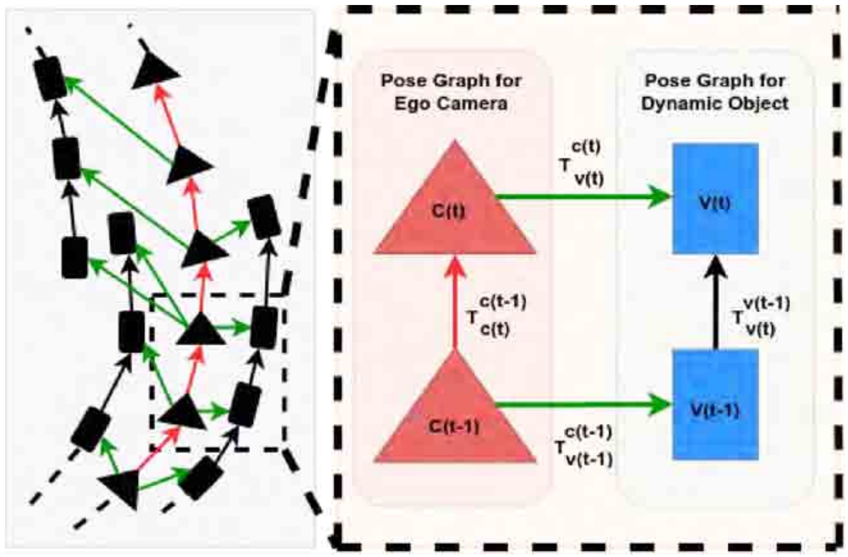


Figure 4.2: An illustration of the pose-graph formulation that resolves *cyclic-consistency* between object and camera across consecutive images. Triangular blocks represent ego-camera while rectangular blocks represent dynamic objects in the ego-camera's scene respectively. Red, green and black edges represent camera-camera, camera-object and object-object edges respectively.

4.6 Exploiting Static Cues from the Environment

We also utilize environment static-cues to improve agent motion by constraining it with respect to lane. We obtain a dense point-cloud set P_l for road plane segregated for each lane. As illustrated in Fig.4.3 we define a unary-edge between an agent 'A' and each point $p \quad P_l$ on the lane as in Eqn. 4.8.

$$\Psi = \Sigma_p(tr_A^W - X_p^W), \;\; (p \quad P_l) \quad tr_A^W - X_p^W \; \leq \tau \tag{4.11}$$

Here, $tr_A^W \quad \mathbb{R}^3$ represents the translation vector in global frame 'W' for an agent node 'A' which could be the ego-vehicle or any dynamic-object in scene. $X_p^W \quad \mathbb{R}^3$ is the 3D depth to the lane-point $p \quad P_l$. Fig.4.3 illustrates how the lane based edges are applied to each instance in the trajectory for an object. To ensure the trajectory behaviour is not lost, we provide a very high confidence to the vehicle-vehicle edges. Since road could introduce variations at longer depths from the object, we introduce threshold τ shown in Eqn.4.11 from each agent instance 'A' such that a unary-edge is formed between 'A' and a lane-point 'p' only if 'p' is within a distance $\tau \quad \mathbb{R}$ from 'A'. The coordinates to all lane-points $p \quad P_l$ are obtained from an architecture which is outside the scope of this thesis.

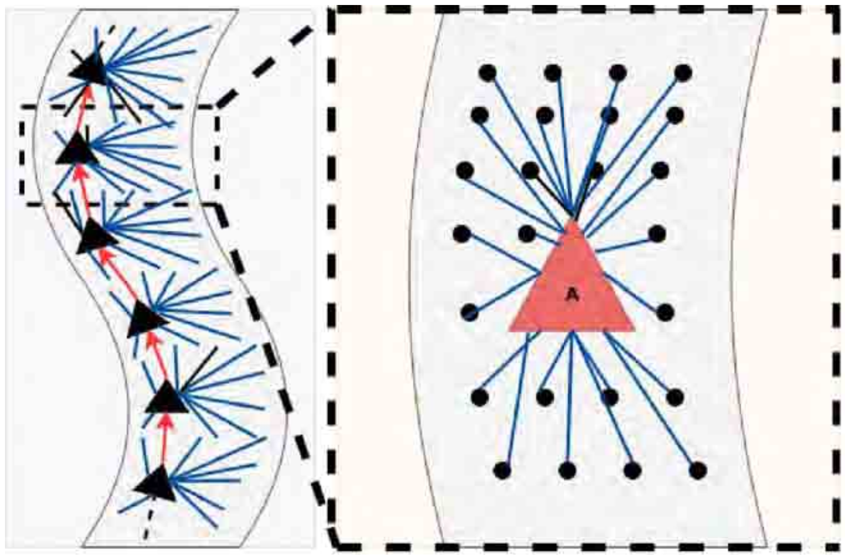


Figure 4.3: An illustration of lane-based pose-graph formulation from Sec. 4.6. Triangle blocks represent agent instances, directed red edges represent agent-agent biary-edges and undirected blue edges represent unary-edges between the agent instance and the stationary lane-points in environment.

4.7 Cumulative Pose-Graph Formulation

Mathematically, we aggregate our pose-graph contraints as a combination of Eqn.4.10 and Eqn.4.11 applied alternatively to on-road trajectory based scenarios. The aggregate pose-graph based costs can thus be represented as Θ where:

$$\Theta = \Upsilon + \Psi \tag{4.12}$$

4.8 Experiments and Results

4.8.1 System Setup

Most of the experiments explained in the following sections have been developed and run using MATLAB version R2019b. This includes preparation and compilation of input and output data as well as plotting and visualization of results and statistics. Pose-graph optimizations are performed using the g2o C++ library [16] (*cf.* 3.3.2, 4.6) and Gaussin Process based time-series predictions from Sec.4.3 are performed using the GPML toolbox [34]. All of the experiments are performed on a system enabled with quadcore Intel i7-5500U CPU with $2.40GHz$ processor.

4.8.2 Dataset

We report experiments performed on a variety of long on-road training sequences from KITTI-Tracking dataset [10] with a number of dynamic traffic-objects in the scene, primarily consisting of cars and minivans. As long as the objects are non-truncated and non-occluded to the ego-camera, our localizations prove reliable irrespective of various possible orientations and maneuvers subjected to the object-motion or the ego-motion.

We evaluate our object localizations with respect to the KITTI-Tracking labels [10] and we treat the GPS/IMU trajectories extracted from the corresponding OXTS files as our ground truth for ego-motion.

4.8.3 Approaches Evaluated

We evaluate the performance of the following approaches:

- *CubeSLAM*[47]: A monocular approach that unifies 3D object detections and and multi-view object SLAM pipelines in a way that benefits each other.
- *Namdev* et al.[25]: A monocular multibody VSLAM approach that obtains motion for dynamic objects and ego-camera in a unified scale. The non tractable relative scale that exists between the moving object and camera trajectories is resolved by imposing the restriction that the object motion is locally linear.
- *Our Batch-Wise Approach with Scale Initialized Odometry from Chapter.3*: A monocular multi-body approach with a batch-wise pose-graph optimization formulation that resolves relationships with dynamic objects as a means of performing SLAM.
- *Our Batch-Wise Approach with Scale Ambiguous Odometry*: In contrast, the ego-camera trajectories here are in their own scale i.e. the scale in which Monocular SLAM trajectories are estimated, which is different from the scale in which the vehicle nodes are initialized.
- *Our Real-Time Approach with Scale Initialized Odometry*: Here the pose graph nodes corresponding to ego vehicle trajectory are scale initialized through the method described in section 3.2.2.2. In other words the camera nodes in the pose graph are always in the same scale as the ego vehicle nodes.

4.8.4 Experiments

4.8.4.1 Real-Time Approach with Scale Initialized Odometry

We run these experiments enabled with Gaussian Process based trajectory predictions from Sec.4.3 and pose-graph optimization on a frame-by-frame basis. We only use object-localizations to bootstrap the pipeline in the first $F = 5$ frames. A major improvement in this mode of experiment from the above batch-based approaches from Chapter.3 and Sec.4.8.4.2 is that these xperiments do not use future trajectory poses to optimize for the current poses. In other words, experiments in Chapter.3 and Sec.4.8.4.2 is more akin to a Multibody SFM framework, while the current effort is a Multibody SLAM framework that uses only the observations until the current time instance.

However, these experiments are similar to those in Chapter.3 such that the odometry estimates used for camera-camera binary-edges are *scale-initialized* from Sec.3.2.2.2 and the confidence parameterization is relatively very high for the same consequently. Given that the Gaussian Process based trajectory predictions may add its own error-model to the vehicle-vehicle binary-edge initialization, they are provided with comparatively lower confidence parameterization. Meanwhile, the camera-vehicle binary-edges are fed with a coupled parameterization from Sec.4.2.2 and Sec.4.2.1 similar to the experiments detailed in Chapter.3 and Sec.4.8.4.2.

4.8.4.2 Batch-Wise Approach with Scale Ambiguous Odometry

Just like in Chapter.3, we again resolve each cyclic-loop for the ego-camera with each vehicle (*cf.* Eqn.4.7) in the scene in these experiments thus going ahead with a very similar formulation as Chapter.3. However, the difference here is that the odometry initialization is provided to the pose-graph structure as *scale ambiguous* odometry from Sec.3.2.2.1. I this mode of experiments, our pose-graph formulation boldly attempts to scale the *scale ambiguous* initialization upto the same scale as that of the other edges which is the *metric scale* post optimization.

Here, since the odometry initialization are at a different ambiguous scale altogether, they are extremely unreliable to trust with providing a direction to the optimization problem. Thus, we provide the least confidence to the camera-camera edges initialized with *scale ambiguous* odometry estimations when compared to the other edges in the formulation. The camera-vehicle and vehicle-vehicle edges follow the same confidence parameterization behavior as that in Chapter.3.

Foreseeing that the above formulation alone may not be sufficient to scale the *scale ambiguous* odometry accurately to the *metric scale*, we take aid from the *lane based constraints* from Sec.4.6 too. These help curb a potentially high lateral drift in trajectories of the ego-camera as well as other dynamic vehicles in the camera's scene.

4.8.5 Qualitative Results

We have obtained very accurate maps for the reported sequences across multiple multi-body dynamic scenarios from Chapter.3. We aim at improving those results while showing much superior perforance in the experiments explained above in Sec.4.8.4.2 and Sec.4.8.4.1.

4.8.5.1 Batch-Wise Version with Scale Ambiguous Odometry

A natural follow up direction to take for us was to test if the same pipeline could handle *scale-ambiguous* odometry initialization from Sec.3.2.2.1. Thus, we performed these experiments, again in a batch-mode as explained in Sec.4.8.4.2 along with lane-based constraints in our pose graph formulation as described in Sec.4.6. The experiments produced very encouraging results as has been illustrated in Fig.4.5. The pose-graph optimizer has done a exceptional job at utilizing known trajectory in *metric scale* of various dynamic vehicles in scene to pull *scale-ambiguous* odometry to the *metric scale*. The initializations whose scale was so off from the *metric scale* that the odometry estimates would remain very close to the world origin itself, have been optimized so efficiently that we obtain close to ground truth performance for the ego-camera estimates at as a result of the pipeline as is evident from Fig.4.6.

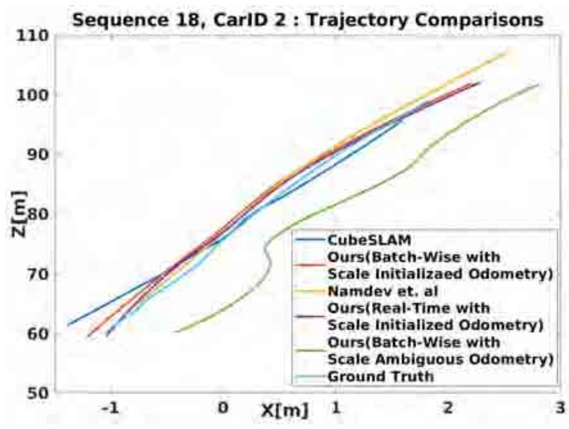
4.8.5.2 Real-Time Version with Scale Initialized Odometry

As an attempt to take a step forward to real-world perception, we also work with a real time version with *scale-initialized* odometry initializations where the *cyclic-consistencies* of the pose-graph structure from Sec.3.3.2 are resolved for each pair of frames sepertely from each other as the data is made available. This pipeline also performs wonderfully with a very encouraging set of resultant visualizations as summarized by Fig,4.7. The performance with respect to ground truth is also commendable as shown in Fig.4.8. The GPML based trajectory predictor module is used here from Sec.4.3 which makes the optimization problem slightly more challenging by introducing its own error model to the system.

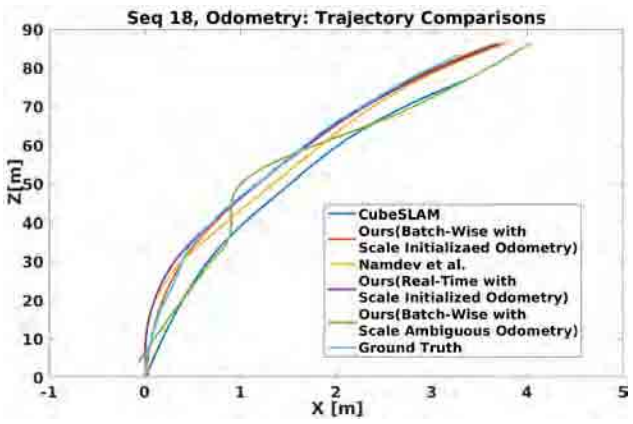
Fig.4.4 shows the performance of all evaluated approaches as introduced in Sec.3.4.2 on a comparative setting for two vehicle instances from out evaluated data. We observe that, while both of our approaches that take *scale-initialized* odometry for input from Chapter.3 and Sec.4.8.4.1 show minimal lateral drift throughout the estimated trajectory, there generally a higher order of lateral drift in the estimated trajectory from our approach from Sec.4.8.4.2 that takes *scale-ambiguous* odometry inputs. However, any lapse in terms of lateral drift is made up for by an improved accuracy along the depth of the camera's perpective where both of the competitors CubeSLAM[47] as well as Namdev*et al.*[25] do not put ahead a great performance. On the whole, based on the qualitative plots from Fig.4.4, we can see the all our approaches show a near ground truth performance in the reported vehicle instances.

4.8.6 Quantitative Results

Table.4.2 and Table.4.1 summarize the quantitative performance of our approach against each other as well as other approaches tackling the same problem from Sec.3.4.2. Clearly, we record superior trajectory estimations with respect to corresponding ground truth. We show substantial improvements in ATE(Absolute Translation Error) over [47] for each of the three vehicles considered along with the ego-car in sequence 18. We also outperform Namdev*et al.*[25] on trajectory estimates for instances of most vehicles recorded in the traffic scene including the ego-camera itself estimations.



(a)



(b)

Figure 4.4: Performance of all pipelines evaluated as summarized in Sec.3.4.2 for two vehicle instances which are Part(a): Sequence 18, Car ID 2 and Part (b): Sequence 19, Ego Car. While blue and yellow trajectories represent trajectory estimates from CubeSLAM[47] and Namdev*et al.*[25] respectively; red, violet and green trajectories illustrated the performance of our three experiments from Chapter.3, Sec.4.8.4.1 and Sec.4.8.4.2 respectively. Ground truth trajectory is plotted as a light blue trajectory.

As was visible in the qualitative results above, we can see that our approach from Sec.4.8.4.2 puts forth a much better performance than all other approaches because of a substantial improvement in the performance factored in by the lane-based constraints from Sec.4.6. Sec.4.8.7.1 also analyses the contribution of lane based constraints when compared to the same pipeline in the absence of lane based constraints. While the trajectory initializations to various vehicle instances in the ego-camera's scene relied on the odometry initializations for substantial improvement post pose-graph optimization in all the experiments that used the *scale-initialized* odometry iniatializations described in Chapter.3 and Sec.4.8.4.1; the batch-based approach with *scale-ambiguous* odometry initializations functions in a different manner such that the *cyclic-consistencies* and *lane-based* constraints are resolved separately in an alternatng fashion. Consequently, both the odometry and the dynamic vehicle trajectory estimations get several chances to correct themselves with respect to the stationary features in the scene as well as re-caliberate the inter-object relationships. This in turn leads to the optimization problem being less reliant on any one initialization sources making itself potentially more effective and robust.

4.8.7 Ablation Studies

4.8.7.1 Impact of Lane-based Constraints in Pose-Graph Formulation

We further make progress in making a thorough analysis as to how significant the lane-based constraints are to the overall performance of the pipeline. We do so by simply comparing the performance of one of our batch based pipelines from Sec.4.8.4.2 before one episode of appication of the lane-based constraints as described in Sec.4.6 with the same after.

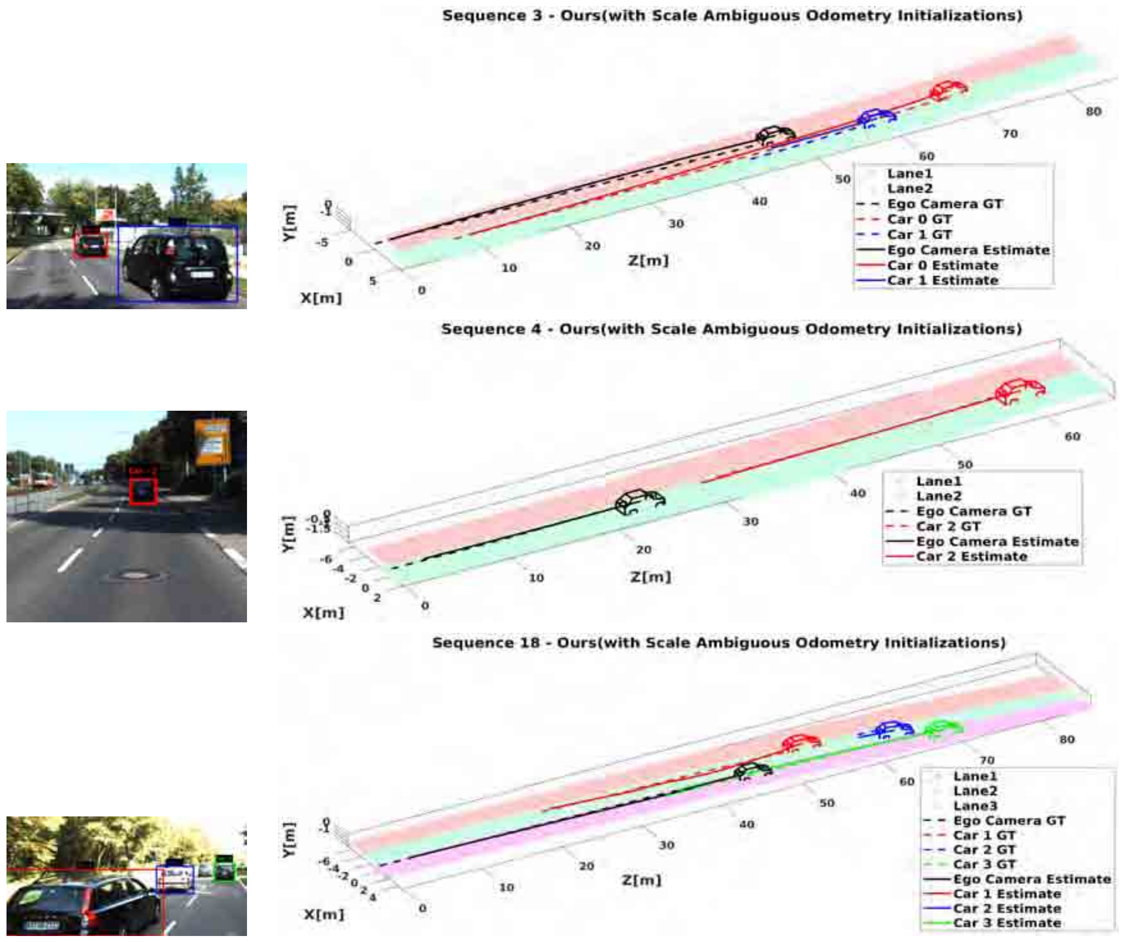
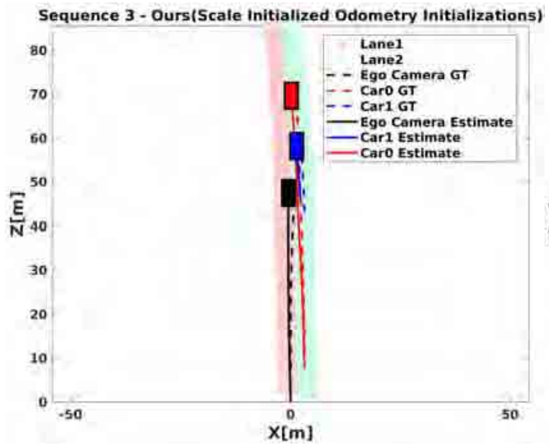
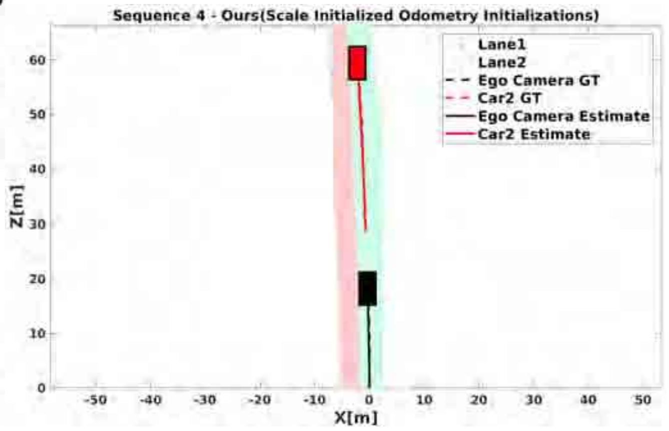


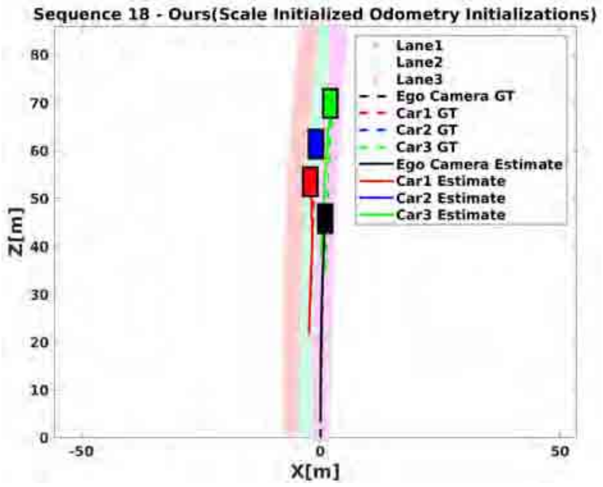
Figure 4.5: Qualitative results on various sequences for **Batch-Wise Approach with Scale Ambiguous Odometry** as explained in Sec.4.8.4.2. *Col 1* shows input images with bounding boxes for vehicles mapped in *Col 2* and *Col 3*. While *Row 1* and *Row 3* illustrate our performance on multi-vehicle traffic scenarios, *Row 2* shows results for a far away vehicle over a long sequence. Ego-vehicle is shown in black whereas the red, blue and green plots show unique vehicles in the scene with corresponding dotted plots showing the ground truths. Note that entire ground truth trajectory is shown at once in the figures whereas our results are up to the instance frame shown in *Col 1*.



(a)



(b)



(c)

Figure 4.6: Qualitative top-view plots for **Batch-Wise Approach with Scale Ambiguous Odometry** from Sec.4.8.4.2. The three plots show the top view version for the same sequences shown in the perspective 3D mapping illustrated in Fig.3.6. The road plane is represented light red, light green and light violet planes whereas the odometry estimates are show as a black trajectory. Other dynamic vehicles in the camera's scene are show as red, green and blue trajectories. Dotted trajectories represent the respective ground truth trajectories.

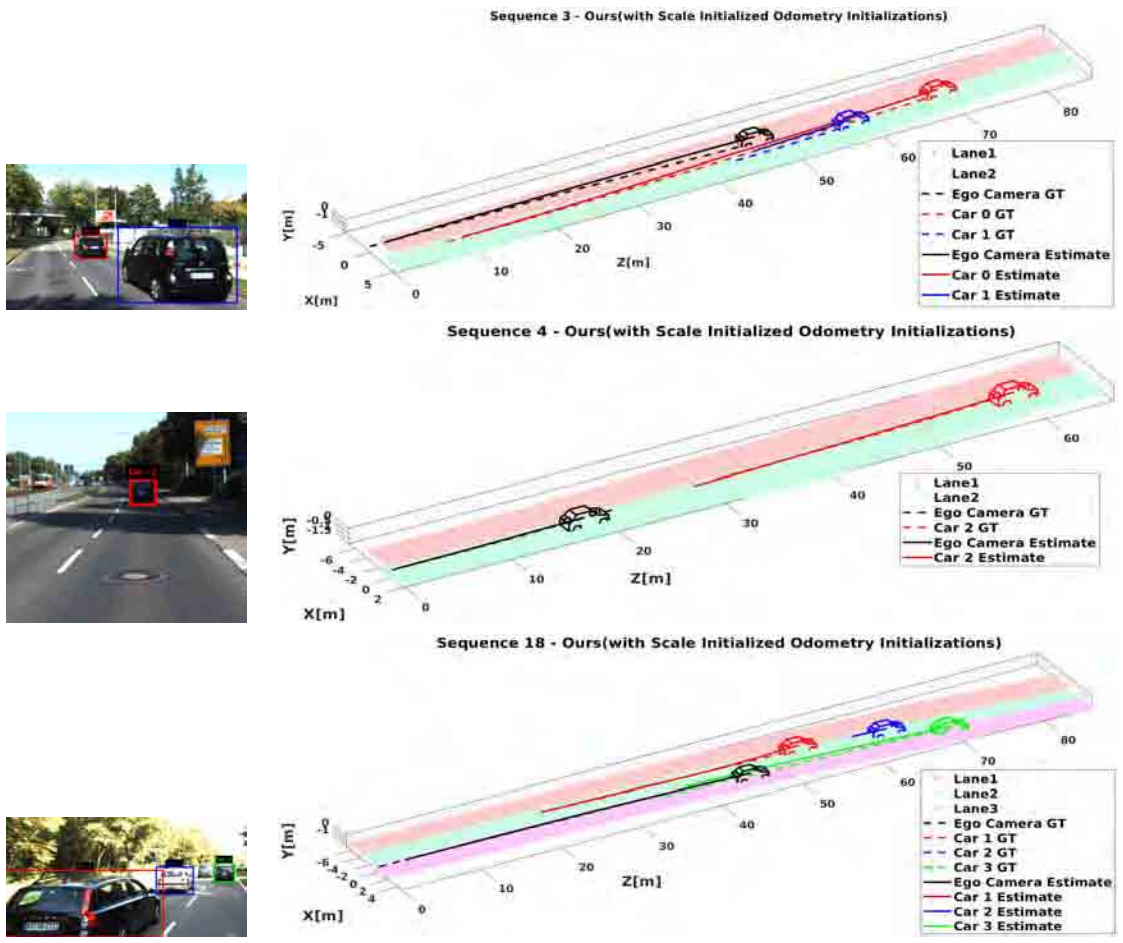
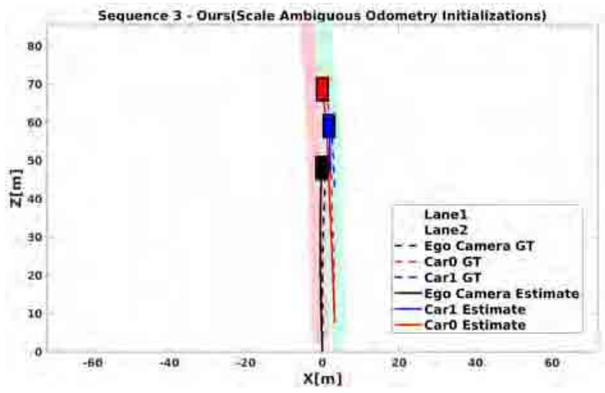
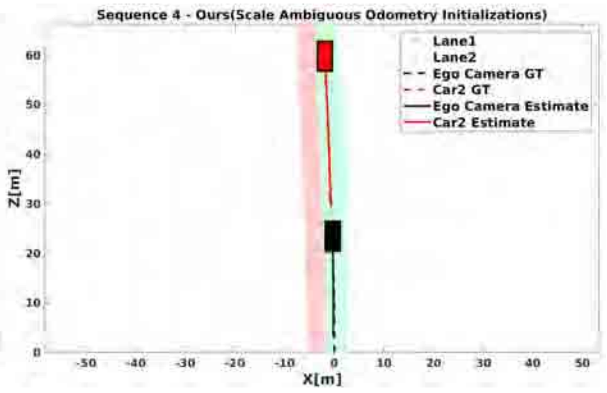


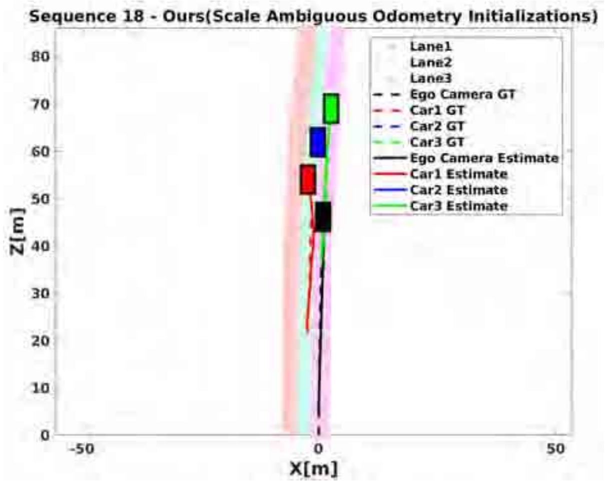
Figure 4.7: Qualitative results on various sequences for **Real-Time Approach with Scale Initialized Odometry** as described in Sec.4.8.4.1. *Col 1* shows input images with bounding boxes for vehicles mapped in *Col 2* and *Col 3*. While *Row 1* and *Row 3* illustrate our performance on multi-vehicle traffic scenarios, *Row 2* shows results for a far away vehicle over a long sequence. Ego-vehicle is shown in black whereas the red, blue and green plots show unique vehicles in the scene with corresponding dotted plots showing the ground truths. Note that entire ground truth trajectory is shown at once in the figures whereas our results are up to the instance frame shown in *Col 1*.



(a)



(b)



(c)

Figure 4.8: Qualitative top-view plots for **Real-Time Approach with Scale Initialized Odometry** from Sec.4.8.4.1. The three plots show the top view version for the same sequences shown in the perspective 3D mapping illustrated in Fig.3.6. The road plane is represented as light red, light green an light violet planes whereas the odometry estimates are show as a black trajectory. Other dynamic vehicles in the camera's scene are show as red, green and blue trajectories. Dotted trajectories represent the respective ground truth trajectories.

Absolute Translation Error (Root Mean Square) in Global Frame (metres)							
Seq No.	3		4	18			Avg Error
Car ID	0	1	2	1	2	3	
Frame length	41	92	149	62	83	141	
CubeSLAM[47]	-	-	-	1.89	2.43	7.17	3.83
Namdev*et al.*[25]	13.81	11.58	11.18	3.77	5.93	3.72	8.33
Ours(Chapter.3)	**1.61**	4.99	2.14	1.29	3.45	2.4	2.65
Ours(Sec.4.8.4.2)	2.20	**2.24**	**1.77**	**1.21**	2.86	**1.23**	**1.92**
Ours(Sec.4.8.4.1)	2.28	2.4	2.14	1.35	**2.31**	1.83	2.05

Table 4.1: Quantitative performance for various dynamic vehicles in the scene of the ego-camera for all evaluated approaches as summarized in Sec.3.4.2 in a static-global frame in a single *metric scale*. The errors are computed as root-mean-squared Absolute Translation Errors averaged throughout across complete trajectories for each of the vehicles instances reported in the table.

Absolute Translation Error (Root Mean Square) in Global Frame (metres)				
Seq No.	3	4	18	Avg Error
Frame length	123	149	141	
CubeSLAM[47]	-	-	2.99	2.99
Namdev*et al.*[25]	11.49	11.12	3.69	8.77
Ours(Chapter.3)	1.96	6.49	**2.27**	3.57
Ours(Sec.4.8.4.2)	**1.96**	**1.89**	2.36	**2.07**
Ours(Sec.4.8.4.1)	1.96	6.43	2.24	3.54

Table 4.2: Quantitative performance for ego-camera trajectories in the scene for all evaluated approaches as summarized in Sec.3.4.2 in a static-global frame in a single *metric scale*. The errors are computed as root-mean-squared Absolute Translation Errors averaged throughout across complete trajectories for each of the vehicles instances reported in the table.

These are performed on our batch-version of experiments with *scale-amibuous* ORB initializations as we observe that this mode of experiments came with a higher degree of potential errors post reolution of *cyclic-consistencies* thus providing a wider opportunity for us to make any quality observations with the lane-based constraints. Moreover, the nature of the error model followed was also dominantly in the form of a lateral shift, which is what the lane-based constraints targets to concentrate on fixing. We show that lane-constraints contribute substantially to the ATE for almost all vehicles which are experimented with, when compared with that before applying lane-based constraints. We summarize our observations in Table.4.4 and Table.4.3.

Absolute Translation Error (Root Mean Square) in Global Frame (metres)							
Seq No.	3		4	18			Avg Error
Car ID	0	1	2	1	2	3	
Frame length	41	92	149	62	83	141	
Before Lane-Constraints	2.91	2.61	2.15	1.32	3.22	**1.19**	2.23
After Lane-Constraints	**2.20**	**2.24**	**1.77**	**1.21**	**2.86**	1.23	**1.92**

Table 4.3: Ablation studies on impact of lane-based constraints on batch-based approach from Sec.4.8.4.2 for trajectories of various instances of dynamic vehicles in the ego-camera's scene across various recorded sequences from our evaluation data.

Absolute Translation Error (Root Mean Square) in Global Frame (metres)				
Seq No.	3	4	18	Avg Error
Frame length	123	149	141	
Before Lane-Constraints	2.26	4.82	2.53	3.20
After Lane-Constraints	**1.96**	**1.89**	**2.36**	**2.07**

Table 4.4: Ablation studies on impact of lane-based constraints on batch-based approach from Sec.4.8.4.2 for various instances of ego-camera trajectories across various recorded sequences from our evaluation data..

4.8.7.2 Impact of Depth-Thresholding in Lane-Constraints

Our experiences with lane-based constraints from Sec.4.6 also got us to introduce a new parameter τ which thresholds the depth from the agent up to which the lane-based constraints are applied. The agent here could be the ego-camera or a dynamic-object from the scene. This comes based on our extensive experimentation with the feature point obtained on ground plane. Our major observation was that, given that considering larger patches of the road along the depth of the camera's perspective may welcome larger degrees of variations in the road plane itself in the form of curves or turns. Thus, we felt picking lane point correspondences from the near vicinity of the agent itself maybe more reliable to ensure better performance of lane-base constraints for that instance of the agent.

We thus introduce the depth parameter τ and experiment with various values for that. We observe that our pose-graph formulation performs the best when thresholded at $\tau = 5m$. We record our observations for various values for τ over a few objects in scene including ego-camera in Table. 4.5 and Table.3.3.

Absolute Translation Error (Root Mean Square) in Global Frame (metres)								
Seq number		3		4	18			Avg Error
Car ID		0	1	2	1	2	3	
Frame length		41	92	149	62	83	141	
Depth	$\tau =$	3.48	**2.50**	2.90	2.70	4.50	1.26	2.89
Parameter	$\tau = 20$	3.35	2.83	2.66	2.64	4.43	1.19	2.85
τ	$\tau = 10$	3.12	2.98	2.25	2.43	4.17	**1.16**	2.68
(metres)	$\tau = 5$	**2.92**	2.96	**1.85**	**2.09**	**3.78**	1.17	**2.46**

Table 4.5: Analysis over depth parameter τ for lane-based optimization formulation in batch-based approach from Sec. 4.6 for trajectories of various instances of dynamic vehicles in the ego-camera's scene across various recorded sequences from our evaluation data.

Seq number		3	4	18	Avg Error
Frame length		123	149	141	
Depth	$\tau =$	2.17	2.38	2.44	2.33
Parameter	$\tau = 20$	2.00	2.31	2.40	2.24
τ	$\tau = 10$	**1.93**	2.12	2.37	2.14
(metres)	$\tau = 5$	1.96	**1.89**	**2.36**	**2.07**

Table 4.6: Analysis over depth parameter τ for lane-based optimization formulation in batch-based approach from Sec. 4.6 for various instances of ego-camera trajectories across various recorded sequences from our evaluation data.

4.8.7.3 Run-time Analysis for the Real-Time Approach with Scale Initialized Odometry

We observe that our pose-graph formulation corresponding to Sec.4.8.4.1 resolves inter-object constraints in competitive time-intervals irrespective of number of objects in the scene. Fig. 4.9 shows how a single and multi-object scenario fare in terms of runtime for each incoming instance. Pose-graph optimizations (*cf.* 4.4) are performed in real time on a quadcore Intel i7-5500U CPU with $2.40GHz$ processor. The pose-graph formulation optimizes for a single multibody-instance with an average time-latency of $16.33ms$. Our formulation provides estimates for each recorded instance thus providing odometry as well as multibody trajectory estimations at identical frame-rates.

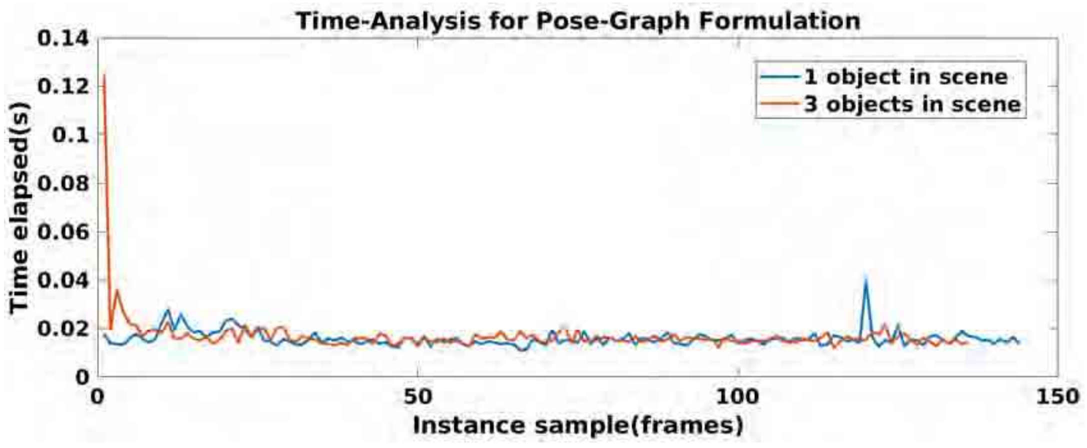


Figure 4.9: Plot illustrating how number of objects in scene do not affect the time-elapsed in our optimization formulation from Sec. 4.8.4.1

Chapter 5

Conclusions

Monocular Multi-body SLAM is indeed a difficult problem due to its ill-posed as it is impossible to triangulate a moving vehicle from a moving monocular camera. This observability problem manifests in the form of relative scale when posed into the Multibody framework. With the arrival of single view reconstruction methods based on Deep Learning, some of these difficulties are alleviated, but one is still entailed to represent the camera motion and the vehicles in the same scale. There have been very few attempts to tackle this problem head on even in the Deep Learning era. Classical methods were able to unify vehicle and camera trajectories for few number of frames while modern methods that involve Deep Learning and projective geometry to compute object pose and shape do so only in the local camera frame as the problem of observability continues to be difficult to resolve.

This paper proposed a simple yet effective two stage solution to the problem. It lifts the camera trajectory to the same scale as object trajectories by making use estimated depth to ground plane feature correspondences. This ensures sufficient initialization for the trajectories to the ego-vehicle and other dynamic participants with respect to a unified frame in metric scale. It further robustifies for the scale and camera and vehicle poses through a multi-body pose-graph optimizer framework that can seamlessly integrate many moving objects. Both incremental and batch formulations were proposed and significant performance gain with prior formulations are established.

We show trajectories of dynamic objects and ego vehicle over sequences of more than a hundred frames in length with high fidelity ATE (Absolute Translation Error). To the best of our knowledge, this is the first such method to represent trajectories in the global frame over long sequences. The pipeline accurately maps trajectories of dynamic participants far away from ego camera and its scalability to map multi-vehicle trajectories is another salient aspect of this work. Future directions include integrating Deep Prediction modules to further improve the mapping of dynamic trajectories.

Related Publications

1. **Gokul B. Nair**, Swapnil Daga, Rahul Sajnani, Anirudha Ramesh, Junaid Ahmed Ansari, Krishna Murthy Jatavallabula, and K. Madhava Krishna. *Multi-Object Monocular SLAM for Dynamic Environments*. In the **31st IEEE Intelligent Vehicles Symposium (IV)** 2020. Accepted.

2. **Gokul B. Nair**, Swapnil Daga, Rahul Sajnani, Anirudha Ramesh, Junaid Ahmed Ansari, Krishna Murthy Jatavallabula, and K. Madhava Krishna. *Mapping Trajectories of On-Road Dynamic Participants with a MovingMonocular Camera: A Multibody Approach*. In **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)** 2020. In Review.

Bibliography

[1] S. Agarwal, K. Mierle, and Others. Ceres solver.

[2] J. A. Ansari, S. Sharma, A. Majumdar, J. K. Murthy, and K. M. Krishna. The earth ain't at: Monocular reconstruction of vehicles on steep and graded roads from a moving camera. In *IROS*, 2018.

[3] S. Avidan and A. Shashua. Trajectory triangulation: 3d reconstruction of moving points from a monocular image sequence. *PAMI*, 2000.

[4] C.-H. Chang, S.-C. Wang, and C.-C. Wang. Vision-based cooperative simultaneous localization and tracking. In *ICRA*, 2011.

[5] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *ICCV*, 1995.

[6] A. J. Davison, I. D. Reid, N. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[7] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *ECCV*, 2014.

[8] A. W. Fitzgibbon and A. Zisserman. Multibody structure and motion: 3-d reconstruction of independently moving objects. In *ECCV*, 2000.

[9] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 2017.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 2013.

[11] R. Girshick. Fast r-cnn. In *ICCV*, 2015.

[12] G. Grisetti, R. Kümmerle, H. Strasdat, and K. Konolige. g2o: a general framework for (hyper) graph optimization. In *ICRA*, 2011.

[13] M. Han and T. Kanade. Multiple motion scene reconstruction from uncalibrated views. In *ICCV*, 2001.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017.

[15] T. He and S. Soatto. Mono3d++: Monocular 3d vehicle detection with two-scale 3d hypotheses and task priors. In *AAAI*, 2019.

[16] R. Kuemmerle. g2o - general graph optimization.

[17] A. Kundu, K. M. Krishna, and C. Jawahar. Realtime multibody visual slam with a smoothly moving monocular camera. In *ICCV*, 2011.

[18] P. Li, T. Qin, et al. Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving. In *ECCV*, 2018.

[19] M. Machline, L. Zelnik-Manor, and M. Irani. Multi-body segmentation: Revisiting motion consistency. In *ECCV Workshop on Vision and Modeling of Dynamic Scenes*, 2002.

[20] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017.

[21] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[22] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 2017.

[23] J. K. Murthy, G. S. Krishna, F. Chhaya, and K. M. Krishna. Reconstructing vehicles from a single image: Shape priors for road scene understanding. In *ICRA*, 2017.

[24] J. K. Murthy, S. Sharma, and K. M. Krishna. Shape priors for real-time monocular object localization in dynamic environments. In *IROS*, 2017.

[25] R. Namdev, K. M. Krishna, and C. V. Jawahar. Multibody vslam with relative scale solution for curvilinear motion reconstruction. In *ICRA*, 2013.

[26] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[27] K. Ozden, Kemal E anld Schindler and L. Van Gool. Multibody structure-from-motion in practice. *PAMI*, 2010.

[28] K. E. Ozden, K. Cornelis, L. V. Eycken, and L. V. Gool". Reconstructing 3d trajectories of independently moving objects using generic constraints. *CVIU*, 2004.

[29] H. S. Park, T. Shiratori, I. Matthews, and Y. Sheikh. 3d reconstruction of a moving point from a series of 2d projections. In *ECCV*, 2010.

[30] P. Parkhiya, R. Khawad, J. K. Murthy, B. Bhowmick, and K. M. Krishna. Constructing category-specific models for monocular object-slam. In *ICRA*, 2018.

[31] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-dof object pose from semantic keypoints. In *ICRA*, 2017.

[32] R. Ranftl, V. Vineet, Q. Chen, and V. Koltun. Dense monocular depth estimation in complex dynamic scenes. In *CVPR*, 2016.

[33] S. R. Rao, A. Y. Yang, S. S. Sastry, and Y. Ma. Robust algebraic segmentation of mixed rigid-body and planar motions from two views. *IJCV*, 2010.

[34] C. E. Rasmussen and N. Hannes. Gaussian processes for machine learning matlab code version 4.2.

[35] C. E. Rasmussen and H. Nickisch. Gaussian processes for machine learning (gpml) toolbox. *Journal of machine learning research*, 2010.

[36] N. D. Reddy, I. Abbasnejad, S. Reddy, A. K. Mondal, and V. Devalla. Incremental real-time multibody vslam with trajectory optimization using stereo camera. In *IROS*, 2016.

[37] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *CVPR*, 2017.

[38] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 2015.

[39] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2013.

[40] S. Rota Bulo, L. Porzi, and P. Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. In *CVPR*, 2018.

[41] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, and S. Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. *arXiv preprint*, 2018.

[42] K. Schindler and D. Suter. Two-view multibody structure-and-motion with outliers through model selection. *PAMI*, 2006.

[43] S. Song and M. Chandraker. Joint sfm and detection cues for monocular 3d localization in road scenes. In *CVPR*, 2015.

[44] R. Vidal, Y. Ma, S. Soatto, and S. Sastry. Two-view multibody structure from motion. *IJCV*, 2006.

[45] S. Wangsiripitak and D. W. Murray. Avoiding moving outliers in visual slam by tracking moving objects. In *ICRA*, 2009.

[46] L. Wolf and A. Shashua. Two-body segmentation from two perspective views. In *CVPR*, 2001.

[47] S. Yang and S. Scherer. Cubeslam: Monocular 3-d object slam. *IEEE Transactions on Robotics*, 2019.

[48] H. Zhan, R. Garg, C. Saroj Weerasekera, K. Li, H. Agarwal, and I. Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *CVPR*, 2018.