Visual State Estimation for Autonomous Navigation

by

Ali Salimzadeh

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

University of Alberta

Abstract

This thesis aims at improving robot perception for autonomous navigation in highly dynamic environments. In the first part of this research, a fixed-frame visual localization method utilizing a fisheye monocular camera is proposed to enhance the navigation accuracy for autonomous mobile robots in dynamic indoor environments (with direct application to warehouse or service robotics). The method develops an optimal variance filter with covariance adaptation for visual state estimation and is able to address challenging scenarios due to full/partial occlusion. In the second part of this thesis, a novel and computationally-eficient visual-inertial dynamic object detection and tracking framework are proposed, using onboard visual and inertial sensors for autonomous navigation in highly-dynamic environments to address challenges of existing localization and navigation methods that heavily rely on the assumption of static features in the scene or use learning-based methods to detect dynamic objects. The novel framework combines prediction over inertial data with the measurements from stereo vision-based state estimation to form a stochastic filter with Bayesian tracking for motion classification. In this pipeline, point cloud clustering, disparity map generation, and consistent tracking have also been conducted for both fixed- and moving-frame scenarios. The proposed frameworks are experimentally validated in several autonomous navigation scenarios in highly dynamic indoor/outdoor environments and in urban settings. The two distinct solutions presented in this thesis, are designed to resolve challenges imposed by the dynamic nature of the environment in-

cluding occlusions and unreliable feature selection for localization. The results of this thesis confirm the reliable, consistent, and real-time performance of the developed frameworks in both fixed frame (i.e., infrastructure-based) and moving frame state estimation using multimodal visual-inertial data. Combining the two solutions proposed in this thesis for networked robotic systems and connected autonomous driving leads to more precise, robust, and eficient autonomous navigation systems that can be used for both indoor and outdoor applications.

Preface

The first part of the methodology and research approach of this thesis (i.e., Chapter 3), which focuses on fixed-frame visual state estimation is presented as a full research paper in the 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE) and later published in the conference proceedings as Augmented Visual Localization Using a Monocular Camera for Autonomous Mobile Robots with authors: Ali Salimzadeh, Neel P. Bhatt and Ehsan Hashemi.

The second part of the research on dynamic object detection using moving visual frames (i.e., Chapter 4) is being prepared for submission to IEEE Transactions on Intelligent Vehicles.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Prof. Ehsan Hashemi, for his invaluable guidance and support, encouragement, and expertise throughout my Master's program, research publication, and delivering research outcomes.

I would also like to thank Prof. Bob Koch, Prof. Dan Sameoto, and Prof. Osezua for serving on my thesis committee and providing helpful feedback and suggestions.

I am grateful to Dr. Neel P. Bhatt and Dr. Arunava Banerjee for their wonderful research and technical mentorship and support which helped me complete my research at the NODE lab.

I would also like to thank my friends and family for their love and support during my studies. Without their support, this academic journey would have been impossible.

Finally, I would like to acknowledge the financial support received from the Natural Sciences and Engineering Research Council of Canada, Alberta Innovates, and Major Innovation Fund (Alberta Ministry of Technology and Innovation).

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Autonomous navigation refers to the ability of a vehicle or a robot to lo-cate itself in any given environment and discern a path that will lead to its desired destination without human intervention. Based on such capabilities, autonomous navigation has found its use in a wide variety of industries, which include but are not limited to transportation, logistics, agriculture, and de-fense. In all of these industries, autonomous robots are required to navigate in remote or highly dynamic environments that are crowded with human work-ers, moving machinery, and other vehicles. Thus, such complex environments with their dynamic nature, require eficient solutions that can enable the au-tonomous agent to perceive its surroundings faster, which can pave the way for a more rapid decision-making algorithm to perform navigation tasks safely. For an optimal execution of this task, the autonomous agent should be equipped with the resources to detect and track dynamic objects and use this infor-mation to navigate within the dynamic environment. In this context, state estimation is commonly used for effective and safe navigation, even with noisy data, since measurement errors are a natural part of all data-gathering pro-cesses, and all sensors exhibit some level of uncertainty regardless of their build quality. A wide range of sensors, including LIDARs, GNSS, cameras (monoc-ular or stereo vision), and inertial measurement units (IMUs) are used to estimate the robot/vehicle states including position, orientation, and longitu-dinal/lateral velocities (in the body frame). In this regard, pre-processing the measured data is essential for removing noise and calibration errors before be-

ing utilized in the state estimation process. Thus, for a variety of autonomous navigation tasks, including localization, mapping, and motion planning, accurate estimation of the vehicle's and the surrounding dynamic objects' states is crucial for safe and timely operation of the vehicle in challenging environments and perceptually degraded conditions. This research (and the thesis) addresses the accuracy and computational challenges of reliable detection and state estimation of dynamic objects in such arduous conditions and proposes and experimentally validates two approaches for fixed- and moving-frame visual state estimation for single and multi-robot settings. More details about the motivation of this research are provided in section 1.1, followed by the problem statement and the scope of the thesis in section 1.2. Lastly, the contributions of both localization and perception parts of this thesis are outlined in section 1.3.

1.1 Motivation

Autonomous navigation in dynamic environments poses significant challenges for localization algorithms. Traditional approaches rely on the assumption of a static scene[31], [77], [78], [84], which yields inaccurate results when features used for localization exhibit motion patterns [47]. Matching the relative poses of these dynamic objects temporally can result in large reprojection errors, introducing uncertainties in the trajectory estimates of autonomous vehicles or robots. To address this issue, it is crucial to detect and account for these motion patterns and selectively remove foreground objects that are not fixed with respect to the environment, such as vehicles and human agents. A methodical approach is required to enable autonomous navigation in highly dynamic environments. Instead of indiscriminately removing all moving objects, a more sophisticated strategy involves detecting and removing dynamic objects based on their motion patterns. By doing so, localization can be performed with reduced motion-induced uncertainties while preserving essential visual information for robust feature and object detection. By detecting and removing dynamic objects based on their motion patterns, autonomous navigation sys-

tems can operate more reliably in complex and highly dynamic environments. This approach enhances localization accuracy and robustness by mitigating the negative impact of independent motion patterns, allowing for more precise trajectory estimations and safer navigation. Furthermore, by selectively removing dynamic objects and focusing on the static background, feature and object detection algorithms can extract meaningful information for map building, motion planning, and decision-making processes in real-time scenarios. Making a reliable navigation solution by developing a stochastic motion-aware perception algorithm is the main motivation of the research presented in Chapter 4.

However in indoor environments, where the operational space of autonomous robots is limited, the utilization of centralized sensing information and computations can greatly benefit autonomous navigation. By deploying visual sensor units mounted on infrastructure, several advantages can be achieved compared to relying solely on onboard sensors. Firstly, the visual measurements obtained from infrastructure-mounted sensors are more accurate due to precise calibration and the absence of motion-induced uncertainties. Additionally, having a bird's eye view of the operational area enables the observation of multiple robotic agents simultaneously, leading to enhanced coordination and reduced costs. Furthermore, stationary sensors are not constrained by the weight and size limitations of onboard computational hardware, allowing for highly scalable solutions. This approach finds application in various domains, including warehouse robotics, service robotics, and automated manufacturing plants. The exploration of this concept is further detailed in Chapter 3, while the associated challenges are discussed in the subsequent section.

The shift in testing conditions and the environment from Chapters 3 to 4 is a strategic move aimed at addressing safety concerns crucial for autonomous navigation. The higher vehicle speed and the presence of multiple moving objects in the surroundings provide an ideal scenario to evaluate the capabilities of the developed state estimation framework. Additionally, the assessment of stochastic state estimation methods in both indoor and outdoor settings enhances the trustworthiness of the proposed approach and its suitability for

ensuring navigation safety.

In this regard, some of the main challenges associated with autonomous navigation in highly dynamic environments (for mobile robots and automated driving systems in urban settings) include:

Sensor uncertainties: Mobile robots with onboard sensory units encounter motion-induced errors. To address these uncertainties, sensor fusion frameworks are developed in Chapters 3 and 4, incorporating prior information about the process in optimal filters with Bayesian tracking to achieve accurate system state estimation despite noisy measurements. In addition, a state estimation framework for localizing multiple agents using a stationary visual sensor is presented in Chapter 3 and could be used for applications with human interaction, such as networked and warehouse robotics. This framework enables safe navigation around independently moving agents by computationally-fast estimation of the robot (and dynamic objects') states which will be utilized for cost map generation using the Edge computing capability of the developed remote sensing units in the NODE lab.

Localization: A model-based approach for robust dynamic object detection, addressing the challenge of accurate localization in dynamic scenes is presented in Chapter 4. By effectively detecting and handling dynamic objects, the proposed method enhances the accuracy of localization and navigation algorithms in dynamic environments.

Motion planning and control: This research aims to improve perception capabilities for reliable and accurate motion planning and execution in dynamic environments, enabling eficient autonomous navigation. Furthermore, in highly dynamic environments, autonomous navigation systems require accurate risk assessment. The proposed framework in this thesis enhances perception capabilities to enable superior risk management, facilitating the development of comprehensive risk-aware planning algorithms for autonomous operations of mobile robots (and vehicles).

1.2 Problem Statement and Scope

The objective of this research is visual-based state estimation for autonomous navigation using fixed or moving stereo vision sensors with computational constraints due to vehicle/robot onboard processors and embedded systems at remote visual sensing units (i.e., infrastructure-mounted sensors). Dynamic objects and tracking their features significantly reduce the accuracy of the visual localization due to erroneous position estimates by the visual odometry optimization program that assumes utilizing fixed features for triangulation and re-projection to calculate translation matrices. This thesis is mainly focused on theoretical and practical aspects of developing a reliable and consistent dynamic object identification framework that can be used by existing visual or visual-inertial navigation pipelines [11], [47], [84]. As a result, two distinct frameworks are proposed and experimentally verified in this thesis to address visual-based state estimation accuracy and computational eficiency challenges in dynamic scenes, each targeting indoor or outdoor environments and perceptually degraded conditions. Furthermore, these two frameworks and continuous (or event-triggered) communication between them (i.e., the infrastructure-mounted sensing unit and onboard estimator) can be used together to improve the perception and reliable navigation of a network of robots in dynamic scenes.

The initial part of this thesis is focused on using an infrastructure-mounted visual sensor for localizing an autonomous mobile robot. This technique is commonly referred to as infrastructure-aided localization. Utilizing a fixed camera to observe the operation of autonomous mobile robots has multiple advantages over the use of onboard sensors. Moving sensors tend to suffer from motion-induced errors and noises. Stationary sensors, on the other hand, avoid these uncertainties by utilizing highly accurate calibration techniques. These calibrations are only done prior to operation and result in more reliable measurements. Moreover, visual sensors that are mounted on the infrastructure, usually have a better point of view and can oversee a large region of the robot's operation. These sensors can also avoid occlusions from dynamic objects since

most of the dynamic activities occur at ground level. Avoiding occlusions can improve the consistency and reliability of visual detection which in turn will provide more accurate localization results. Therefore, infrastructure-aided localization is the focus of the first part of this thesis. This technique can also be combined with the use of onboard sensors for increased visibility, a higher level of safety, and added redundancy. The details of this work are explained in detail in Chapter 3.

The next part of this thesis introduces a real-time technique for detecting dynamic objects from onboard visual-inertial sensors. Outdoor environments introduce more challenges such as a higher number of object classes for tracking, uncontrolled light, and weather conditions, and more sporadic motion patterns compared to indoor environments. Consequently, this part of the thesis is developed for autonomous driving applications in busy city streets. First, a broad range of dynamic objects are detected visually based on AI-generated bounding boxes. Then, their motion is analyzed in a virtual 3D environment created in the algorithm. Classifying the motion patterns of these objects around the ego-vehicle, results in a binary selection of visual objects. Finally, the dynamic objects can be removed from the initial visual frames, to produce completely static frames that are useful for simultaneous localization and mapping applications.

1.3 Contributions

The contributions of this research are listed in the following under two main research Themes: infrastructure-aided localization and dynamic object detection with moving camera frames for autonomous navigation purposes. The contributions for the first part are summarized as:

- Designing an uncertainty-aware state observer with covariance adaptation: To address detection and depth estimation deficiencies caused by occlusions or algorithmic failure, an uncertainty-aware state observer is developed with covariance matrix adaptation model based on the prior information about the sensor noise level. Additionally, the

states of the system are augmented by the robot's speed, which increases the accuracy and robustness of the estimations.

- Devising a data-driven model to estimate robot acceleration: In order to predict the accelerations of the mobile robot, a neural network is designed and trained on a moving horizon of position measurements captured by the remote visual sensing unit. The output of this model is then used in the state observer to ensure accurate estimation of the robot pose with stable and bounded estimation error. Asymptotic stability of the estimation error dynamics is also theoretically analyzed and proved through observability analysis of the optimal variance filter.

The contributions of the dynamic object detection framework, which classifies the motion patterns of different objects in a dynamic scene with full/partial occlusion (for autonomous driving applications), are summarized hereby:

- Visual-inertial velocity estimation: To address the estimation drift problem associated with inertial integration techniques, a visual-based velocity estimation node is developed to track static features and estimate the optimal temporal velocity of the ego-vehicle (or ego-robot) using a recursive least squares.

- Geometrical vector closure model for motion classification: A geometrical-based motion analysis (which benefits from ego-motion compensation) is designed to estimate the motion of the surrounding objects with temporal tracking.

- Bayesian motion tracking for multiple objects: a novel continuous dynamic probability distribution is devised to enable tracking of binary motion states with measurable confidence levels. This motion probability distribution is updated incrementally using a Bayesian recursive inference that gets its input from the raw measured motion states.

Chapter 2

Literature Review

In this section, the background and challenges of visual-based state estimation and existing methods for identifying dynamic objects for simultaneous localization and mapping (SLAM) are provided to better highlight the challenges mentioned in section 1.2. A comprehensive understanding of the existing challenges will also help to give a clear notion about the motivations of this research. First, a technical description of the literature for dynamic object detection is provided. Then, background information on the tools used in those studies is presented in the following sections.

Visual-inertial navigation using monocular or stereo vision has been well established in the literature [31], [77], [78], [84] by tracking features using on-board sensory data and optimizing the corresponding re-projection error (bundle adjustment). However, the presence of dynamic objects and the resulting occlusions are the main challenges for existing state-of-the-art SLAM approaches which result in estimation drift [47]. For warehouse or service robots working in an indoor and dynamic environment, this issue can be resolved by localizing the robot independently with an infrastructure-mounted camera and possible communication with the robot. As a result, the accurate infrastructure-aided localization information could be used for reliable navigation, motion planning, and controls utilizing wireless communication between the robot/vehicle and a base station [19], [70], [127].

Visual localization algorithms can benefit from fish-eye monocular vision due to the increased field of view and reduced operational and computational

costs for the navigation of mobile robots in both indoor and outdoor environments. For instance, color segmentation from single fish-eye camera frames is conducted in [19] to localize a robot. Frame differencing is utilized in [85] for images from a camera installed on the ceiling. However, perceptually degraded lighting conditions and shadows are challenging for these methods. A robust state estimator is developed in [27] using a series of cameras that detect infrared beacons installed on the robot. A single wide-angle ceiling-mounted camera, with unknown intrinsic and extrinsic parameters, is used in [58] for mobile robot navigation robust to model uncertainties. Localization approaches that use sensor/camera networks (e.g., [111]) have been shown to be precise, but they require an extensive calibration process. To summarize, mobile robot localization using infrastructure-aided monocular vision involves two main challenges: detection failure due to occlusions and sudden lighting changes; and frame distortion due to the wide-angle lens and noisy visual data.

Dynamic object detection and tracking algorithms in scenes captured by moving visual frames, such as those obtained from onboard monocular or stereo vision systems in autonomous vehicles or mobile robots, heavily depend on the selection of sensors and data representations. Various sensors such as light detection and ranging (LIDARs) [64], [73], [97], [120], event cameras [26], [126], and stereo or RGB-D cameras [23], [115], [118] can be used on the robot platforms based on the kinematic/dynamic constraints. Regardless of the choice of sensor types, the processing algorithms can be classified into two main categories including model-based and fully data-driven methods. Although data-driven methods are segmented in a separate category, some model-based approaches also take advantage of machine learning, especially in visual data processing. However, the use of learning-based tools is limited in model-based methods to the extent that the core of their work is centered around a mathematical, geometrical, or probabilistic system representation.

Model-based approaches are superior to end-to-end learning-based methods, as they provide stability (due to bounded state trajectories and consequent bounded estimation error coming from the inherent characteristics of stable dynamical models) and prediction based on bounded inputs, and take

advantage of engineering intuition in their design. For example, in [80], depth images are used to generate U-depth and V-depth maps, which estimate the states of obstacles and demonstrate safe navigation with static obstacles. Similar techniques are employed in [59] and [93], where U-depth maps are utilized to detect and track obstacles and represent them as 3D ellipsoids. Furthermore, dynamic obstacles are detected using depth and U-depth maps in [118], and the results are combined with an occupancy map for navigating dynamic environments. Other methods focus on detecting and segmenting dynamic obstacles in 2D image planes to improve localization robustness and construct fully static maps. For example, dynamic image patches are segmented through motion-compensated frame differencing and motion tracking using particle filters and Maximum-a-posterior (MAP) estimator on vector quantized depth images as seen in [104]. Moreover, 3D dynamic features are detected by tracking the relative motion of 3D map vertices in [18] and removing length-varying correlations from graph optimization. On the other hand, a combination of object segmentation with optimization to reduce short-term photometric reprojection errors of tracked objects is done in [3] to create a dynamic mask, aimed at improving localization accuracy.

Fully data-driven methods, on the other hand, try to predict dynamic objects based on training over 3D point clouds or semantics. For example in [121], dynamic objects are semantically segmented on RGB-D images, and predictions are filtered based on their moving consistency. In Another method [109], a logistic classifier is built on a binary feature map created from a voxelized point cloud grid. The training of the classifier is performed on the KITTI dataset, with positive class examples provided by bounding box tracklets of dynamic obstacles. On the other hand, the method described in [99], utilizes the region proposal network from YOLO [89] to predict the location and orientation of bounding boxes. This approach combines motion and appearance features to extract moving objects directly and can complement existing dynamic object extraction methods. MODnet [98] is another approach that learns to extract moving objects by leveraging motion and appearance features. This method can be used to enhance existing dynamic object extraction methods by

directly extracting objects based on learned features. Another method that is based on multi-modal background subtraction is provided in [81]. Multi-modal background subtraction is performed by classifying image data as foreground (dynamic obstacles) or background. This is achieved by using a per-pixel Gaussian mixture model estimated on a background without dynamic obstacles. LIDAR point clouds are reprojected into the camera's field of view and masked as foreground or background using the existing correspondence between pixels and LIDAR points. This enables the application of background subtraction methods on image data to perform binary classification of LIDAR points. The aforementioned fully data-driven models are challenged by corner cases that are not included in the learning dataset. In turn, obtaining big datasets (for training) to fully cover real-world scenarios is time-consuming, operationally expensive, and not safe, especially for autonomous driving at the capacity limit or high speed. Consequently, a model-based approach is proposed (and experimentally verified in various urban driving scenarios) in chapter 4, which detects possibly moving objects and tracks them using a Bayesian recursive scheme in 3D to ensure consistent classification of their motion over time. On the other hand, a learning-aided optimal variance fil-ter is designed in chapter 3 for position and heading estimation of mobile robots using a stationary fisheye camera. This approach avoids visual oc-clusions and motion-induced errors and provides more accurate localization results utilizing the prior information about the robot and the environment in an uncertainty-aware optimal filter. This solution is suited for indoor and outdoor applications, such as service robots in dynamic environments, auto-mated storage with mobile robots in warehouses, and automated driving in urban settings [16], [45], [55], [119]. As mentioned previously, existing visual or inertial-based navigation solutions are reaching their performance limits in highly dynamic environments due to vision-based feature/point tracking challenges in uncertain scenes with dynamic objects, perceptually degraded conditions, or their growing complexities in model-based approaches in the presence of wheel slippage and tire force non-linearity, impacting estimation error and update frequency in real-time.

To design a visual-based state observer, different techniques such as object detection, feature extraction, and stereo matching are required as they play a vital role in harnessing both model-based and data-driven approaches [6], [11], [78], [79]. In the following sections, a review of these tools will be presented, along with an exploration of available solutions in the literature.

2.1 Machine Vision

Machine vision, also known as computer vision, combines artificial intelligence and computer science to extract meaningful information from digital images or video sequences. Its objective is to enable computers to perform complex visual tasks, mimicking and surpassing human visual capabilities. In the field of robotics, machine vision plays a crucial role in tasks such as depth perception, object detection, segmentation, tracking, and scene understanding. The key to understanding visual frames is the identification of distinctive and informative points, known as features or keypoints, within an image. Early approaches to feature extraction, such as those in [75] and [21], evaluated gradients and curvature analysis to identify points of interest. However, these methods had limitations. More effective strategies were introduced, such as using the Hessian matrix to capture curvature information [67], calculating eigenvalues from the inverse of the auto-correlation matrix, and introducing indicators for size and similarity [28]. In [37], a new criterion called the cornerness value was proposed based on the determinant of the Hessian matrix which helped to increase the accuracy of keypoint detection. These advanced methods significantly improved feature detection capabilities by identifying distinctive points in images based on local characteristics.

Multi-scale interest operators, such as those discussed in [7], detect features at multiple scales and match them across scales. However, this approach has some limitations when the scale difference is too large or the scale ratio is not known. A more advanced method is to use scale-space theory, as described in [62], which convolves the original image with a 2D Gaussian function at different scales. By analyzing the local extrema in scale-space using the Laplacian of

Gaussian (LoG), scale-invariant features can be obtained. The Scale-Invariant Feature Transform (SIFT), introduced in [66], approximates the normalized LoG using the Difference of Gaussian (DoG) and performs sub-pixel localization by fitting a local quadratic model to the extrema. SIFT is well-known for its effectiveness in matching images with scale changes and can tolerate some afine distortion. The research proposed in [72], introduced a scale selection mechanism to the Harris corner detector, preserving points where the LoG is an extremum. This research also introduced iterative refinement for both scale and position. These methods leverage scale-space analysis and LoG or DoG operators to detect scale-invariant features, enabling robust feature detection and matching across different scales.

Feature description involves converting features into a descriptor space that allows for easy distinction and matching. The purpose of designing descriptors is to achieve invariance against various transformations like lighting changes, rotation, and afine distortions. There are two main types of descriptors: floating point and binary descriptors. Floating point descriptors offer better discriminability but are computationally more expensive, while binary descriptors are suitable for applications with limited computational resources [14]. The process of determining descriptors involves several steps: transformation, aggregation, normalization, and dimension reduction[116]. Transformation operations, such as calculating gradients or Haar feature responses, are applied to preserve and amplify local patterns. Aggregation combines information from small regions within the feature support window, enhancing robustness against noise and limited transformations. Normalization ensures that values are within a fixed range, reducing the impact of absolute response and improving robustness against lighting changes. Dimension reduction techniques like principal component analysis (PCA) are used to compress the representation and eliminate redundancy in high-dimensional descriptors [14].

Classic descriptors, including SIFT [66] (Scale Invariant Feature Transform) and SURF [4](Speeded-Up Robust Feature), have been widely used in computer vision. SIFT calculates gradients within a feature's support window, applies Gaussian filtering, and aggregates gradients in aligned grids. SURF

uses Haar wavelet response and aggregates the gradients in square grids. These descriptors have built upon previous advancements, providing robust and distinctive representations of image features for various computer vision applications. On the other hand, binary descriptors offer an alternative approach to feature description by comparing pairs of pixel values and representing the results as binary strings. BRIEF [9](Binary Robust Independent Elementary Features) compares pixel values at different positions within a smoothed support window. ORB [91](Oriented BRIEF) extends BRIEF by incorporating orientation estimation and using a greedy search to select optimal pixel positions. BRISK [56](Binary Robust Invariant Scalable Keypoints) determines the orientation of keypoints in scale-space and performs gray value comparisons in concentric circles. These binary descriptors provide eficient feature representations that facilitate faster processing and matching of features.

2.2 Stereo Matching

In this section, an overview of different stereo matching algorithms is provided to establish the different possible methods that could be used for visual reprojection. Stereo matching involves finding corresponding pixels in a pair of stereo frames that represent the same feature in the 3D environment. The disparity, which is the horizontal distance between matching pixels in rectified images, is measured to create a disparity map that contains disparity values for all image points and can be utilized for estimating scene depth [79]. Stereo algorithms can be classified into two main categories based on their global and local disparity measurement techniques [95]. Local algorithms, such as block matching and gradient-based methods, focus on matching points within a local window and use metrics like correlation, rank, and intensity difference [8]. Examples of local algorithms include SAD correlation metric [76], SMP algorithm [20], and ZNCC integrated with a neural network [5]. Feature-based methods handle depth discontinuities and texture uniformity by utilizing features like edges [8], including segmentation matching [125] and hierarchical feature-based methods [107]. Global optimization-based methods incorporate

smoothness assumptions and perform iterative disparity computation, achieving accurate results at the cost of computational demand. Examples of global optimization-based methods include dynamic programming [110], graph cuts [113], and belief propagation [102], [114]. Handling occlusions is addressed in methods that utilize techniques like disparity consistency and disparity uniformity [69].

In recent years, deep learning techniques have revolutionized the field of stereo vision, leading to significant advancements in stereo matching algorithms. Various architectures have been proposed to improve performance in this domain. For instance, MC-CNN-acrt [123] utilizes a convolutional neu-ral network (CNN) to compute matching costs for disparity or depth maps, achieving superior performance compared to handcrafted methods. A multi-scale CNN structure [13] preserves relational information between patches and demonstrates improved accuracy. DispNet [71] is an end-to-end network based on an encoder-decoder architecture, while GC-Net [49] utilizes Siamese convolution and constructs a 4D cost volume for matching cost computation. EdgeStereo [101] integrates an edge detection sub-network and achieves state-of-the-art results, while HITNet [106] is a real-time stereo matching network that omits a cost volume and utilizes multi-resolution initialization and geometric propagation. These deep learning-based stereo matching algorithms have significantly improved accuracy and eficiency, pushing the boundaries of stereo vision capabilities.

2.3 Object Detection

Object detection is a challenging task in computer vision, requiring the identification and localization of objects within an image. Early models suffered from slow and inaccurate performance because of their reliance on hand-crafted features. However, convolutional neural networks (CNNs) introduced by AlexNet, revolutionized this field and led to its widespread applications in various domains, such as self-driving cars and medical imaging. Object detection extends object classification by not only recognizing objects but also providing

their coarse localization through bounding boxes. It is a supervised learning problem and relies on large labeled datasets for training and evaluation on benchmarks. Key challenges include intra-class variation, where objects of the same class can vary in appearance due to occlusion, illumination, pose, viewpoint, etc. The large number of object categories poses a challenge, requiring more annotated data. Additionally, the eficiency of object detection models is important, especially with the prevalence of mobile and edge devices. The Pascal Visual Object Classes (VOC) challenge [24], [25], ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[92], Microsoft Common Objects in Context (MS-COCO)[61], and Google's Open Images dataset[53] are widely used for evaluating object detection algorithms. Object detectors are evaluated using various criteria, including frames per second (FPS), precision, recall, and mean Average Precision (mAP). Precision is calculated based on the Intersection over Union (IoU), which measures the overlap between the predicted bounding box and the ground truth. A threshold is set to determine if the detection is correct, and True Positive and False Positive classifications are assigned based on the IoU value. False Negative occurs when an object present in the ground truth is not detected. Precision represents the percentage of correct predictions, while recall measures correct predictions relative to the ground truth. Average precision is computed for each class, and the mean average precision (mAP) is used as a single metric to compare performance across detectors.

Some major milestones in backbone designs for object detection networks include AlexNet[52], VGG[100], GoogLeNet[105], ResNet[39], and ResNeXt [117]. AlexNet introduced the concept of using multiple convolutional kernels and achieved high accuracy in the ImageNet challenge. VGG focused on network depth and utilized smaller filters to reduce parameters while maintaining accuracy. It demonstrated superior performance in image classification and localization tasks. GoogLeNet addressed the computational complexity of deep neural networks by introducing the locally sparse connected architecture, using multiple Inception modules for feature extraction. It achieved high accuracy while being faster than previous models. On the other hand,

ResNet addressed the performance degradation issue by increasing network depth by introducing skip connections, enabling the training of deeper networks. ResNeXt improved accuracy without increasing complexity by using inception-like ResNeXt modules and considering cardinality as a third dimension. It achieved higher accuracy with fewer hyperparameters compared to similar-depth ResNet architectures. CSPNet reduced computational resources while maintaining accuracy by separating feature maps and creating different paths for gradient flow. This approach improved computation unit utilization and memory footprint. These advancements in backbone architectures have significantly contributed to the development of robust and eficient object detection networks.

Object detectors can be categorized into two main types: two-stage and single-stage detectors. Two-stage detectors employ a separate module for region proposal generation, followed by classification and localization in the second stage. These detectors take longer to generate proposals, have complex architectures, and lack global context. In contrast, single-stage detectors perform object classification and localization in a single pass using dense sampling and predefined boxes or keypoints. They achieve real-time performance, have simpler designs, and outperform two-stage detectors in terms of speed.

The first two-stage detector was the Region-based Convolutional Neural Network (R-CNN) [30]. It uses a class-agnostic region proposal module with CNNs to convert detection into a classification and localization problem. R-CNN passes an image through the region proposal module, which produces 2000 object candidates using Selective Search. These candidates are then propagated through a CNN network to extract feature vectors. The feature vectors are passed to class-specific SVMs to obtain confidence scores, and bounding box regression is used to predict the object's location. SPP-Net [35] introduced Spatial Pyramid Pooling (SPP) layers to handle images of arbitrary size and aspect ratio, improving the flexibility of the network. Fast R-CNN [29] improved the speed and eficiency of the two-stage detection pro-cess by introducing a RoI pooling layer and an end-to-end trainable system. Faster R-CNN [90] further enhanced the region proposal generation by intro-

ducing a fully convoluted network as a region proposal network (RPN) and utilizing anchor boxes and shared convolution layers. Feature Pyramid Network (FPN)[60] improved the detection accuracy by constructing high-level semantic features at multiple scales. R-FCN[17] shared computations within the network and used position-sensitive score maps and convolutional layers for localization. Mask R-CNN [38] extended Faster R-CNN to include a branch for pixel-level instance segmentation, achieving better accuracy. These advancements in two-stage detectors have significantly contributed to the progress of object detection.

In contrast to two-stage detectors, single-stage detectors take a different approach to object detection. They reframe the problem as a regression task rather than relying on region proposals. YOLO (You Only Look Once)[87] is an example of a single-stage detector that directly predicts bounding box attributes and object classes for each grid cell in an image. YOLO divides the image into a grid and predicts multiple bounding boxes and confidence scores for each cell. It achieved high accuracy and real-time performance, although it had limitations in localizing small or clustered objects and handling multiple objects within a cell. Another single-stage detector, SSD (Single Shot MultiBox Detector)[63], achieved comparable accuracy to two-stage detectors like Faster R-CNN while maintaining real-time speed. SSD utilized additional auxiliary structures and a VGG-16 backbone to handle different object scales and aspect ratios. It employed techniques such as default boxes, Jaccard overlap matching, hard negative mining, and data augmentation for effective training. Although SSD initially struggled with small object detection, this issue was addressed by incorporating improved backbone architectures like ResNet. Single-stage detectors like YOLO and SSD have played a significant role in real-time object detection tasks.

Moreover, YOLOv2 [89] improved upon the original YOLO algorithm by replacing the GoogLeNet backbone with DarkNet-19, incorporating techniques like Batch Normalization, and using learned anchor boxes for improved recall. YOLOv2 offered a balance between speed and accuracy and provided flexibility in model selection. YOLO9000 extended YOLOv2 to predict 9000

object classes by combining classification and detection datasets using a hierarchical structure called WordTree. YOLOv3 [88] introduced incremental improvements over YOLOv2, replacing the feature extractor with Darknet-53, incorporating data augmentation, multi-scale training, and logistical classifier. However, it had lower accuracy compared to other state-of-the-art detectors. Finally, YOLOv4[6] incorporated various data augmentation techniques, regularization methods, and network enhancements to improve training and inference, resulting in a fast and easy-to-train object detector.

2.4 State Estimation

State estimation combines mathematical models with limited measurements to determine the internal state of a system. By comparing model predictions with measured outputs, errors are corrected, resulting in robust estimates that account for model inaccuracies and measurement noise. This enables accurate analysis and task management in stochastic systems. Kalman filters [46] are powerful state estimators that adapt to non-stationary environments and provide optimal estimates by minimizing mean squared error. They recursively update state estimates based on noisy measurements and consist of time update and measurement update equations, projecting the state estimates forward and incorporating new measurements to improve the estimates. The Kalman filter, renowned for providing estimates that are both unbiased and possess minimum variance, is not without its limitations. These limitations stem from a variety of factors, each influencing the filter's performance and applicability in certain contexts and include poor observability, numerical instability, and blind spots [94]. These limitations can be addressed through sensor changes, supplemental data acquisition, and careful handling of covariance matrices.

The Extended Kalman Filter (EKF) linearizes the equations around the current mean and covariance to apply the Kalman filter algorithm for nonlinear systems or measurement models. The EKF consists of a time update and a measurement update step. In the time update, states are projected based on

the nonlinear process model, and their Jacobian matrix is calculated. State estimates and covariance matrices are updated using the linearized process equation. In the measurement update, the Kalman gain is computed using the linearized measurement equation and error covariance. State estimates are updated based on the difference between the actual measurement and the predicted measurement. The error covariance is updated accordingly. While the EKF allows for nonlinear models, it has limitations [2] such as potential bias, inaccurate error covariance estimation, challenges in finding Jacobian matrices, sensitivity to initial state estimates, and the need for parameter tuning.

In the survey conducted in [44], three types of unscented filtering and nonlinear estimation algorithms were explored: the Unscented Kalman Filter (UKF), Iterated Unscented Kalman Filter, and Unscented Particle Filter. The UKF algorithm, based on linearization using Taylor series expansions, utilizes unscented transformation to represent random variables using a set of deterministically selected sample points called sigma points [43]. These sigma points accurately capture the mean and covariance of the random variable, even in the presence of nonlinearity. The UKF algorithm involves calculating sigma points, performing time and measurement updates using these points, and computing the estimated state. Another study in [54] compared various modifications of Kalman Filters for nonlinear systems, including the Central Difference Filter and UKF, with the Extended Kalman Filter and Iterated Extended Kalman Filter. The UKF algorithm was found to outperform the Extended Kalman Filter in terms of accuracy and computational eficiency, offering a robust approach to nonlinear state estimation problems.

Chapter 3

Fixed Frame Visual State Estimation

Navigation of autonomous vehicles in unknown environments relies heavily on accurate localization which refers to the determination of position and orientation of the vehicle. Accurate localization is crucial for safe decision-making and precise navigation. To achieve this, various sensors such as cameras or range finders like LIDARs and Radars are used to capture the shape of the environment; this is usually the first block of simultaneous localization and mapping (SLAM) algorithms. If the localization is done accurately, the subsequent navigation including mapping, path planning, and control can be done with higher certainty. Cameras are a common choice for localization because of their higher resolution and lower cost compared to LIDARs and Radars. Cameras are also capable of capturing intensity and color information from the scene, which provides additional information about surrounding objects along with geometry. However, cameras are sensitive to changes in lighting conditions and are unable to capture depth information in 2D images. Therefore, additional sensors are used alongside cameras, including LIDARs or inertial measurement units (IMU), to improve the accuracy and reliability of the localization.

Pose refers to the specific position and orientation of an object in a given space. In the case of a vehicle, it refers to its exact location and orientation within its environment. Global optimization of vehicle poses entails minimizing the error term associated with each estimated pose to enhance the accuracy

of tracking the entire trajectory. However, this technique introduces increased run-time and computational complexity, which can pose challenges in real-time navigation scenarios. Additionally, optimization algorithms may suffer from local minima problems, where it converges to a sub-optimal solution. Another method that identifies previously visited locations and uses this information to improve the total error term of a pose graph is loop closure. However, loop closure works best in environments with little change during the navigation session and can struggle with dynamic objects in the environment. The use of inertial sensors or wheel odometry is the third technique that provides additional information about the vehicle's motion. Nevertheless, incorporating these sensors into the SLAM algorithm can add complexity and cost to operations. In general, the choice of sensors and algorithms depends on the specific requirements of the application and the available resources.

The second approach for localization in SLAM algorithms involves using fixed sensors within the environment to observe the autonomous vehicle. This method is known as infrastructure-aided localization and has shown to be significantly more accurate than onboard sensors since the fixed sensors benefit from meticulous calibration processes and do not suffer from motion-induced errors that are often present in onboard sensors. However, they have some limitations regarding their field of view, which means that multiple sensors are required to cover a large environment. Additionally, external localization can be challenging in dynamic environments, where objects can move and cause occlusions or changes in the environment.

Using a hybrid sensory system created by both onboard and external sensors can produce more accurate and robust results. The fusion of the two sensory systems will enable the external sensors to provide a reference for the onboard sensors, reducing the uncertainty and improving the accuracy of the pose estimation. Conversely, onboard sensors have the capability to acquire a more intricate and nuanced perception of the immediate vicinity of the vehicle. This capability proves advantageous, particularly in the context of generating comprehensive and detailed environmental maps. This hybrid system also has a more robust design since it relies on two independent sensory units that

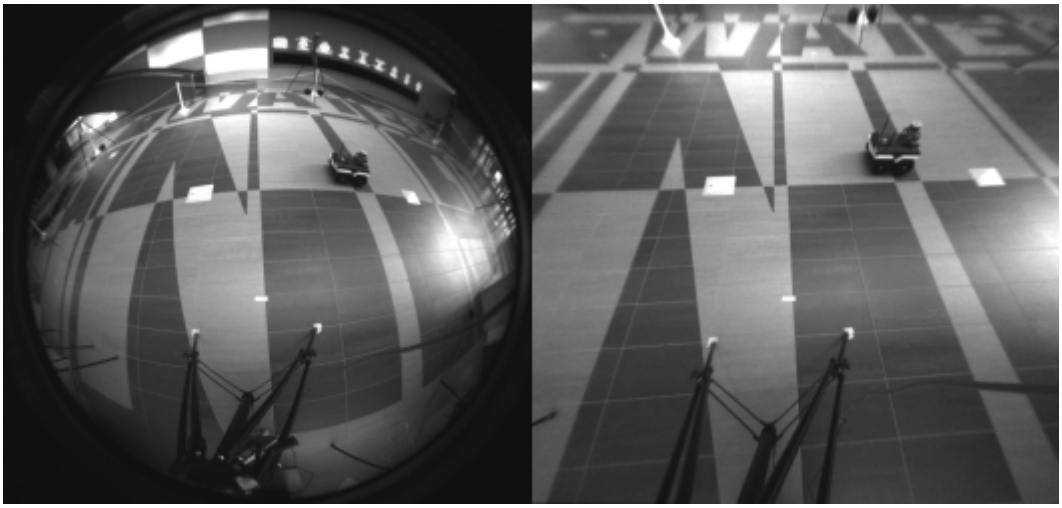
help with its continuous functioning even in the case of a failure in one of the sensory systems.

Thus, the focus of this chapter will be on infrastructure-aided localization using fish-eye monocular vision. The following section provides information about an augmented state estimation framework in which the depth is estimated from an undistorted image to generate a point cloud in a detection-informed region of interest (ROI) for robot localization. This measurement is then used in an uncertainty-aware state observer with adaptive covariance allocation to deal with noisy visual measurements at the limits of the field of view with intermittent occlusions by other dynamic objects.

3.1 2D Perception

Fisheye lenses are commonly used in indoor/outdoor monitoring systems to maximize the field of view of monocular vision. However, these lenses introduce distortions and errors in the image since the field of view gets significantly warped to fit inside a flat circular frame. To address this issue, a fisheye model is used to undistort the frames, as shown in Fig. 3.1. This fisheye model is used to rectify the distorted frames so that they can be accurately used for object detection and depth perception. The parameters for this function are obtained through camera intrinsic calibration using a multi-point correspondence algorithm and a checkerboard, as explained in Appendix A. This calibration process ensures that the camera parameters are accurately estimated, which is essential for the accurate undistortion of the frames.

In order to identify the robot within the 2D undistorted image frame, a customized YOLOv4 object detection model is trained using an exclusive dataset acquired from the fisheye camera within the designated testing environment. The resultant bounding box from this detection process is then employed to isolate the pixels associated with the robot. This pixel set is subsequently projected into a 3D point cloud to facilitate three-dimensional localization. Notably, the generation of the raw point cloud is confined to the pixels enclosed within the YOLO bounding box, which is denoted as the



(a) (b)

Figure 3.1: Visual node image: a) Raw monocular fisheye image b) Undistorted image using the model presented in the Appendix

region of interest (ROI). By narrowing down the search scope within the 3D point cloud, this approach effectively reduces computational demands, distinguishing itself from methods that employ clustering across the entire point cloud dataset[15], [22], [83], [124]. This methodology facilitates more eficient point cloud processing, encompassing the accurate estimation of the robot's pose, which is subsequently employed for the development of an uncertainty-aware state observer. A more comprehensive description of the point cloud extraction procedure is presented in the next section.

Overall, the custom YOLOv4 object detector provides a fast and eficient way to detect the robot in the 2D undistorted image frame, while the bounding box-informed ROI for point extraction reduces the computational cost of point cloud processing. This approach enables accurate pose estimation, which is essential for the development of the uncertainty-aware state observer. To estimate the depth from visual data, traditional methods of fusion between monocular vision and LIDAR/radar sensors [15], [32], [74], [82] or stereo vision [41], [103], have been well investigated in the literature. However, recovering depth from monocular camera frames is recently made possible due to artificial intelligence and also through the utilization of advanced computation hardware. Monocular depth estimation will assist in reducing the complexity

of the system since no sensor fusion or calibration is needed for the single-camera used. In this chapter, MiDas [86] neural network, which has been used in literature to recover a dense map of the environment from a polarimetric mono-camera [96], and in SLAM to recover a dense and globally consistent 3D structure of the environment [65], is utilized to estimate a disparity map from undistorted monocular camera frames. Using MiDas for depth estimation from monocular camera frames has several advantages. It eliminates the need for any additional sensors including LIDARs or more cameras which were previously needed to recover depth. Monocular depth estimation will also be more feasible for a wider range of applications, including mobile and handheld devices. Additionally, the use of a neural network allows the system to be trained on large datasets, improving its accuracy and reliability.

The obtained disparity is then used to recover the depth for each pixel on the image coordinate, and eventually for point cloud generation for the ROI inside the bounding box. The recovered depth is used to calculate the position of the robot in the world frame $\{w\}$. The re-projection of point $p^i = [u^i, v^i, 1]^\top$ in the image frame $\{i\}$ to the point $p^c = [p_x^c, p_y^c, p_z^c]^\top$ in the camera 3D frame $\{c\}$ is obtained by $p_z^c \cdot p^i = K p^c$, where K is the intrinsic camera matrix including the focal lengths f_x, f_y and the principal point coordinates c_x, c_y as in Eq. (3.1). To transform the point cloud into the world frame fixed to the indoor testing environment, affine transformations ${}^wR_c$ and ${}^wP_c$ corresponding to the camera orientation and position in $\{w\}$ are employed to define a homogeneous transformation ${}^wT_c \in R^{4\times4}$ at the time instant k (in discrete-time) as

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \; {}^wT_{c,k} = \begin{bmatrix} {}^wR_{c,k} & {}^wP_{c,k} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{3.1}$$

Following the alignment of the robot point cloud $\{Pcl_k^w\}$ in $\{w\}$, an outlier rejection is conducted. This is required to eliminate unnecessary points (on the ground or far features) from the depth estimator that uses the point cloud within an ROI.

3.2 Point cloud filtering and drift compensation

Visual point cloud generation from a single camera frame produces two types of outliers due to depth uncertainty or occlusions. The first type of outliers corresponds to pixels on the ROI rectangular bounding box that are on the ground. These points are removed by using a proper height threshold which crops the ROI point cloud. The second type of outliers is created around the edges of objects which creates a sudden gradient in the recovered depth map. These edges will be stretched and have a lower density because of the huge gradient in the depth map. Filtering these points is conducted based on a minimal neighboring points threshold $\bar{n}$ (i.e., $\{Pcl^{w}_{f,k}\} = \{Pcl^{w}_{k}\} \setminus S_f, S_f = \{i \in \{Pcl^{w}_{k}\}, N_k(i) < \bar{n}\}$) that removes the points with fewer than $\bar{n}$ neighbors. This two-step process ensures that the ROI point cloud only contains points that belong to the vehicle. Since the robot has a symmetrical body, it is possible to represent its position using only one point. The geometrical center of the filtered ROI point cloud is selected to be the point representation of the robot's position, denoted as $\tilde{p}_k$.

Another type of uncertainty in the point cloud generation arises from drifts in the estimated drift from the MiDas neural network which happens because of the excessive blurring effect that is induced in the fisheye images after rectification. To resolve this issue the characteristics of the error have been studied using recursive least squares. The best fit for the position errors measured against the actual position values (ground truth) obtained from the Vicon system is found to be a fourth-order polynomial ($g(\cdot)$). This polynomial function calculates the position correction vector $e_k = [e_{x,k}, e_{y,k}]^{\top}$ from the measured Euclidean distance of the robot from camera frame $\{c\}$. By using the estimated correction vector elements e_x, e_y from the least squares, the states $\tilde{p}_k = [\tilde{p}_{x,k}, \tilde{p}_{y,k}]^{\top}$ measured from the geometrical center of the point cloud (i.e., $\overline{Pcl}^{w}_{f}$) are corrected to compensate the depth drift as described in Eq. (3.2) and Eq. (3.3), where $\tilde{p}_x$ and $\tilde{p}_y$ are the corrected measurements at time instant k. The estimation of the correction vector and re-projection is

provided in Algorithm 1.

$$p_{x,k} = \tilde{p}_{x,k} + e_{x,k} \tag{3.2}$$

$$p_{y,k} = \tilde{p}_{y,k} + e_{y,k} \tag{3.3}$$

Algorithm 1: Visual-based state vector correction

Input : Raw images in the camera frame ($I^c_{raw,k}$)
${}^wT_{c,k}$ and model Param. $\bar{z}, \alpha_{x,i}\alpha_{y,i}$
Output: Robot state vector $p_k = [p_{x,k}, p_{y,k}]^\top$

1 while $k \geq 0$ do
2 $I^c_{u,k} \leftarrow$ Fisheye undistortion on $I^c_{raw,k}$
3 Extract the robot bounding box and ROI with Yolo;
4 Estimate the NN-based disparity map $I_{d,k}$;
5 Re-project the point cloud $\{Pcl^c_k\}$ using Eq. (3.1);
6 $\{Pcl^w_k\} = {}^wT_{c,k} \cdot \{Pcl^c_k\}$;
7 Geometrical filter $\bar{f}$: $z^w_{i,k} \geq \bar{z}, N_{i,k} \geq \bar{n}, i \in \{Pcl^w\}$
8 $\{Pcl^w_{f,k}\} \leftarrow \bar{f}(\{Pcl^w_k\})$;
9 $\bar{p}_k \leftarrow \overline{Pcl}^w_f := E\{Pcl^w_f\}$;
10 $e_k = \hat{g}(\hat{\alpha}, p_k)$, $\hat{\alpha}(t) = \hat{\alpha}(t) + K(t)[g(t) - \hat{g}(t)]$,
11 RLS: $\arg\min[g - \hat{g}(\alpha, p)]$;
12 $p_k = \tilde{p}_k + e_k$;
13 end

$\alpha^j_x, \alpha^j_y, j \in \{1, \ldots, 5\}$ are the parameters of the distortion correction error function $g(\alpha^j_{x,y}, p)$, and $\bar{z}$ is the threshold for ground points removal in the point cloud cluster within the ROI. The resulting positions coming from algorithm 1 are noisy and thus need to be processed using an optimal filter to remove uncertainties and noises. The next chapter provides additional details about an Uncertainty aware Kalman Filter method that has prior knowledge about the error of the system based on its states.

3.3 Uncertainty-Aware Visual Localization

The system's measured states tend to be affected by noise due to several factors. These include errors introduced by monocular depth estimation, uncertainties related to visual detection, and the common incidence of occlusions,

especially prominent in environments with high levels of dynamic motion. To deal with these uncertainties, one solution is to take the motion model of the robot into account to make the total trajectory of the robot more feasible. This motion model is used in a state observer to compensate for the uncertainties and noises that are present in the measurement (p_k). To simplify the motion model of the robot which is a 4-wheeler in this case, a constant acceleration motion model with adaptive acceleration forecasting is employed which updates based on the measurements obtained from the visual node. This motion model is then fused with the visual node in a Kalman state observer that benefits from adaptive covariance tuning to help by estimating a more reliable trajectory for the robot regardless of the level of noise present in the measurements or the environmental disturbances that are common in dynamic workplaces. The adaptive covariance tuning is designed based on the level of noise present in the measurements when compared to the ground truth and works independently in the operation phase. The discrete-time uncertain model for the state estimation is

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k + \varrho_k, \\ y_k &= C_k x_k + v_k, \end{aligned} \tag{3.4}$$

where the state variable $x_k = [p_{x,k}\ p_{y,k}\ v_{x,k}\ v_{y,k}]^T \in R^4$ includes the position (p_x, p_y) and velocity (v_x, v_y) states which are in the x and y direction of the camera frame (c) respectively. Process uncertainties on position and velocity states represented by $\varrho_k \in R^4$ and visual-based measurement noises $v_k \in R^4$ are bounded. The input $u_k = [a_x,\ a_y]^T$ is the translational accelerations in the longitudinal and lateral directions from the visual node (monocular camera) and is obtained by a learning-based forecasting method using position and speed gradients (refer to section 3.5). The state and input matrices are:

$$A = \begin{bmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ B = \begin{bmatrix} T_s^2/2 & 0 \\ 0 & T_s^2/2 \\ T_s & 0 \\ 0 & T_s \end{bmatrix}, \tag{3.5}$$

where $T_s \in R$ is the sample time between two consecutive frames. It is assumed in the design that the output is exactly the same as the measurements,

thus the output matrix is $C = I_{4\times4}$. It is assumed that $E\varrho_k\varrho_{k\,\square} \leq Q_k$, $E\ \nu_k\nu_{k\square} \leq R_k$. Process and measurement noises are assumed to be uncorrelated i.e., $E\ \varrho_k\nu_{k\square} = 0, \square k \square N$. The assumption of the constant acceleration is valid between two consecutive frames since the system is operating at a high update rate (10 Hz) and the dynamics of the system are substantially slower when compared to the data acquisition rate. Nevertheless, the acceler-ation is updated at every time step (i.e., $T_s = 100$ ms) using a learning-based forecasting method which will be described in the next subsection.

Moreover, the motion model helps with reliable tracking of the robot and localization in case of occlusion in dynamic environments with human presence. These occlusions are common in factories and warehouses where humans and robots are working symbiotically together to ensure higher productivity and safer workspace conditions. Also, in the outdoor environment (i.e. on the streets) all types of dynamic objects are around an autonomous vehicle that could cause occlusions and interfere with the infrastructure-mounted localization sensors. In these cases, a motion model acts as a memory to store the past behavior of the vehicles or mobile robots and provides information about their pose even when sensory detection nodes are failing.

3.4 Stability of the state estimator

The stability of the estimation error dynamics of the Kalman observer, with covariance adaptation for the augmented visual system, is studied in this section. Due to the blurring effect that the rectification step has on the image, the reliability of the depth estimation decreases as the distance between the robot and the fisheye camera frame {c} increases. To address this, an uncertainty-aware adaptive covariance allocation scheme is utilized for the discrete-time observer discussed in section 3.3. Adaptivity in the selection of the process and measurement covariance matrices enables the algorithm to switch its reliance on each one of them based on the state of the system. More specifically, when the robot is close to c and detections are consistent without any occlusions, visual measurements are reliable; thus, the measurement covariance should be

selected such that state estimation relies heavily on the accurate sensor measurements to avoid wrong results. However, as the robot gets far from the camera and occlusions happen more frequently, the noise in the visual detection increases. In this case, the algorithm should be capable of switching its reliance to focus more on the motion model of the robot. The motion model updates based on a moving horizon of position measurements, which increases its robustness to intermittent noises and detection failures. By relying more on the robust motion model of the robot, the Kalman observer can accurately estimate the correct and smooth trajectory of the robot even in edge cases where the visual node provides noisy or faulty measurements.

For the automatic and smooth adaptation of the process and measurement covariance matrices, a transition function is designed based on the Euclidean distance of the robot from the frame {c}. States of the system can be measured through the visual node which is associated with the ”measurement covariance” matrix. On the other hand, states can be predicted with the motion model which is used in the observer alongside the ”process covariance” matrix. Switching the reliance of the state observer is done by increasing or decreasing the values inside each of these covariance matrices. For example, in the case that the robot is close to the camera, the measurement covariance matrix is selected to have smaller values when compared to the process covariance matrix. However, for the case where the visual node is noisy or faulty due to occlusions, the measurement covariance should have greater values compared to the process covariance matrix.

Remark 1 The stability of the state observer relies on the observability of the state space model that was designed for the system, as observability is a suficient condition for the implementation of an optimal variance filter (such as a Kalman). The observability matrix for the system Eq. (3.4) with the system matrix components in Eq. (3.5) can be written as [108]:

$$O_n := [\xi_1, \xi_2, \ldots, \xi_n]^\square$$
$$\xi_{i+1} = \xi_i A + \dot{\xi}_i \xi_1 = C, \tag{3.6}$$

The observability condition is checked and satisfied (i.e., rank(O_4) = 4) for the discrete time-invariant system Eq. (3.4).

Lemma 1 The system Eq. (3.4), with known initial states and covariances, is uniformly detectable.

Proof 1 By definition, the pair $[A_k, C_k]$ in the augmented linear discrete-time system Eq. (3.4) is uniformly detectable if there exists $\bar{\beta} \in R^+$, $0 \leq \beta \leq 1$ and $\forall n_2 \geq 0$, such that

$$\eta^T O(n_2, n_1)\eta \geq \bar{\beta}\eta^T\eta, \tag{3.7}$$

whenever $||\psi_{n_1+\varepsilon,n_1}\eta|| \geq \beta||\eta||$ for some η, n_1 [1]. The observability grammian is denoted by O. The condition Eq. (3.7) necessitates Eq. (3.8) holds for some $\tilde{\beta}$:

$$O = \sum_{k=n_1}^{n_2} \psi_{k,n_1}^T C_k^T C_k \psi_{k,n_1}, \; O(n_2, n_1) \geq \tilde{\beta} I > 0, \tag{3.8}$$

where, $\psi_{i,j} = \psi_{i,i-1}\psi_{i-1,j}$, $\psi_{i+1,i} = A_i$ are the state transition matrices for $i \geq j$. The state matrix for the developed uncertainty-aware visual system is a function of the sampling time, which is constant. For transition and switching between the error covariances (as will be described in subsection 3.6) for far robot positions with respect to {c} (i.e., small ROI), uniform detectability is suficient for bounded error covariance and asymptotic stability of the optimal variance filter. The condition Eq. (3.8) on the observability grammian O holds for the developed (augmented) visual localization system, with known initial covariances. As a result, Eq. (3.4) is uniformly detectable.

Theorem 1 The state observer for the system Eq. (3.4), with the (augmented) state variables in x, bounded system and output matrices, and known initial state/covariance, has bounded error covariance.

Proof 2 With $[A_k, C_k]$ uniformly detectable as in Lemma 1, the Kalman filter error covariance is bounded (see Lemma 5.1 in [1]) and there exists a bounded sequence L_k such that $x_{k+1} = (A_k - L_k C_k)x_k$ is exponentially stable for known initial state and covariance.

The intuitive expression of Theorem 1 is guaranteeing bounded estimation error for the robot pose using the uncertainty-aware stat observer which is subject to switching due to covariance adaptation for far/close features.

3.5 Input estimation

The state space model of the observer discussed in section 3.3 takes linear acceleration inputs in both x and y directions. However, the robot in this application is autonomous which means that the inputs are generated by the robot's perception and controller units. Thus, the infrastructure-mounted sensor does not have any information about the motion inputs to the robot. To be able to reproduce the acceleration inputs, the only source of information is the position measurements which are coming from the camera. For this purpose, a learning-based forecaster is designed to estimate the acceleration inputs for the motion model discussed in Eq. (3.4) using the position mea-surements $p_{x,k}, p_{y,k}$. The input to the network is a moving horizon N_h of the robot positions:

$$p^h_{x,k} = \{\hat{p}_{x,L}, \ldots, p_{x,k}\}, \; p^h_{y,k} = \{\hat{p}_{y,L}, \ldots, p_{y,k}\},$$

where, $L = k - N_h + 1$. The estimated accelerations for both longitudinal and lateral directions are updated with every corrected position measurement p_k at time step k from the visual node utilizing the estimated positions $\{\hat{p}_{x,L}, \ldots \hat{p}_{x,k-1}\}$ throughout the horizon N_h. The number of neurons in each

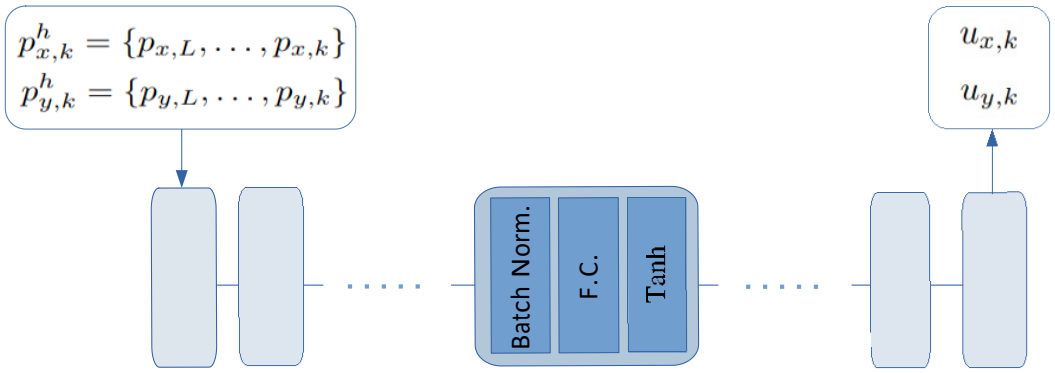


Figure 3.2: Learning-based forecaster for input estimation in the motion model Eq. 3.4. The output of the learning forecaster, which takes input from a moving horizon of the position measurements $p_{x,k}, p_{y,k}$, will be used as the acceleration in-put for the uncertainty-aware motion model. All 15 intermediate blocks have batch normalization and fully connected tanh layers.

layer drop, linearly from the number of elements in the moving horizon (N_h),

to one neuron which is the output acceleration in either longitudinal/lateral directions. The model has been trained on an extensive data set of position measurements from the fisheye camera and the actual accelerations coming from a precise motion capture camera system installed in the environment (Vicon). This helps with making the forecaster robust to the different sources of uncertainties present in the visual measurements captured by the fisheye camera.

After each Tanh layer, a batch normalization layer has been used to regularize the model and increase the learning rate [42]. To estimate the acceleration (which is input to the motion model of the uncertainty-aware observer), every batch is normalized using $\hat{u} = (u - E\{u\})/(\sqrt{var\{u\} + \epsilon})$. Regularization of all layers helps in different ways. The first benefit is improving convergence during training by normalizing the input of each layer and reducing the risk of vanishing gradients. The next benefit is a faster convergence rate which can be beneficial, especially for large data sets and deep neural networks. Also, batch normalization creates a more generalized network that can learn all the excitations present in the data set and produce more stable results. Finally, batch normalization can make the network less sensitive to the initial choices for the weights and biases in each of the neurons.

The idea behind this approach is to leverage machine learning to estimate acceleration inputs to the robot which are dificult to measure externally. Once the accelerations are estimated they can be incorporated into the state space model in Eq. (3.4), which describes how the position and velocity of the robot change over time. This allows for a more accurate representation of the system by taking the dynamics of the robot into account. Since the model is trained on ground truth acceleration measured by precision motion capture sensors, the resulting position estimates of the state space model will also be more precise.

3.6 State estimator

The discrete-time Kalman observer provides the following prediction (with correction) to estimate the robot states, i.e., position and speeds, defined by $\hat{x}_{k+1|j} \triangleq E\{x_k \mid y_j\}$ using a sequence of measurements y_j:

$$\hat{x}_{k+1|k} = A_k \hat{x}_{k|k-1} + B_k u_k + L_k(y_k - C_k \hat{x}_{k|k-1}), \tag{3.9}$$

where the optimal gain is $L_k = A_k \bar{P}_{k|k-1} C_k^{\top} (C_k \bar{P}_{k|k-1} C_k^{\top} + \bar{R}_k)^{-1}$ and error covariance $\bar{P}_{k+1|k} \triangleq \mathrm{cov}\{x_{k+1} - \hat{x}_{k+1|k}\}$ forms a discrete time-varying Riccati equation Eq. (3.10) for both zero and non-zero state initialization $\hat{x}_{0|-1} = E\{x_0\}$ and covariance initialization $\bar{P}_{0|-1} \triangleq \mathrm{cov} x_0 = E\{(x_0 - \hat{x}_{0|-1})(x_0 - \hat{x}_{0|-1})^{\top}\}$:

$$\bar{P}_{k+1|k} = A_k \bar{P}_{k|k-1} A_k^{\mathsf{T}} + \bar{Q}_k - K_k C_k \bar{P}_{k|k-1} A_k^{\top}. \tag{3.10}$$

The Kalman gain and error covariance does not depend on the measurements, even for the time-varying case, but only on the noise statistics. The estimation error is defined by $e_{k+1|j} \triangleq x_{k+1} - \hat{x}_{k+1|j}$, which yields:

$$e_{k+1|k} = (A_k - L_k C_k) e_{k|k-1} - L_k v_k + \varrho_k. \tag{3.11}$$

To address sensor noise/uncertainty dependency on depth due to the blurring effect after undistortion, and small ROI for far distances, process and measurement covariances $\bar{Q}_k, \bar{R}_k$ are switched between two different modes based on an error distance $\tilde{d} \triangleq d_k - \bar{d}$ (with the depth d_k obtained in section 3.1. A and a specific depth threshold d) as in $Q_k = \bar{Q}_d[\frac{1-\gamma_Q}{2}\tanh(sd) + \frac{1+\gamma_Q}{2}]$ and $R_k = \bar{R}_d[\frac{1-\gamma_R}{2}\tanh(sd) + \frac{1+\gamma_R}{2}]$, in which γ_Q, γ_R are the transition coeficients, s is the transition smoothness parameter and Q_d and R_d are the default (diagonal) process and measurement covariance matrices, respectively. For $d_k \triangleq \bar{d}$ the optimal state estimator in the sense of error covariance, relies more on the motion model rather than visual-based position measurements p_k.

The uncertainty-aware state estimation with the input acceleration prediction is provided in Algorithm 2.

Algorithm 2: Augmented visual state observer

Input : Corrected visual-based position states p_k and horizon $p^h_{q,k} = \{\hat{p}_{q,L}, \ldots, p_{q,k}\}, q \in \{x, y\}, L = k - N_h + 1$

Output: States $[\hat{p}_{x,k}, \hat{p}_{y,k}, \hat{v}_{x,k}, \hat{v}_{y,k}]^\top$

Initialize the observer with $\hat{x}_0 \in E\{x\}$ and $P_0 \in E\{(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^\top\}$

while $k \geq 0$ do

  if $k < N_h$ then

    $y_k = [p_{x,k}, p_{y,k}, 0, 0]^\top$;

    $u_k = [0, 0]^\top$;

  else

    $\bar{p}_{q,k} \leftarrow$ stable first-order filter w. τ_x, τ_y on $p_{q,k}, q \in \{x, y\}$;

    $y_k = [p_{x,k}, p_{y,k}, \hat{p}_{x,k}, \hat{p}_{y,k}]^\top$;

    Input estimation

    Learning-based acceleration input u_k estimation using $p^h_{x,k}, p^h_{y,k}$;

  end

  Uncertainty-aware covariance adaptation

  Update $\bar{Q}_k, \bar{P}_k$ based on depth;

  Time update: $x_{k+1} = A_k x_k + B_k u_k$;

  Optimal variance measurement update:

  $L_k = A_k \bar{P}_{k|k-1} C_k^\top (C_k \bar{P}_{k|k-1} C_k^\top + \bar{R}_k)^{-1}$;

  Estimate $\hat{x}_{k+1|k}$ from Eq. (3.9);

end

3.7 Summary

To summarize, a novel approach was proposed in this chapter for localiz-ing a mobile robot in highly dynamic indoor environments using a monocu-lar infrastructure-mounted fisheye camera. The visual sensor measurements, which are inherently noisy, were fused with a constant acceleration motion model using an uncertainty-aware Kalman filter. The novel aspect of this filtering technique lies in its covariance matrix dynamic tuning scheme, which effectively reduces the overall uncertainty of the state estimation by incorporating prior knowledge about the robot's motion and sensor characteristics. By leveraging the visual information captured by the single low-cost camera, the proposed method demonstrates promising results in terms of localization accuracy and robust trajectory estimation. The fusion of visual and motion information enables the algorithm to mitigate the uncertainties associated with

sensor measurements and generate reliable trajectory estimates. An important advantage of utilizing infrastructure-mounted sensors is their large and unobstructed field of view. Additionally, the wide field of view of the fisheye camera lens used in experiments enables overseeing multiple robots within a large indoor environment. This visual sensor serves both as a surveillance system and a localization node of the robotic operation. Moreover, using a camera to detect multiple autonomous robots in an indoor environment provides the possibility to decrease costs and increase the scalability of the operations. Furthermore, having a centralized and fixed sensor unit enables the use of more powerful hardware, thereby increasing the computational capacity required to manage a greater number of mobile agents.

Chapter 4

Dynamic Object Detection

Autonomous navigation systems rely on precise perception of the environment to make informed decisions regarding trajectory planning. Visual sensors, such as cameras, are commonly employed to capture environmental information however, the presence of dynamic objects and scene changes introduces challenges that significantly affect the accuracy of visual perception algorithms. Tracking the features of dynamic objects can lead to errors in visual localization due to the assumption of fixed features for triangulation and re-projection in visual odometry optimization. As a result, existing SLAM algorithms[31], [65], [77], [78] often neglect dynamic objects or exclude potentially moving objects altogether to mitigate localization errors. Nevertheless, such approaches prove inadequate in highly dynamic scenes where dynamic objects dominate the visual frames. To enable autonomous driving in such environments, accurate detection of dynamic objects and scene segmentation becomes imperative. Therefore, the aim of this research is to develop a visual-inertial state estimation framework for autonomous navigation in dynamic environments, leveraging onboard sensors while addressing the computational constraints imposed by the vehicle or robot's onboard processors and embedded systems. The proposed framework aims to enhance visual-based state estimation accuracy and computational eficiency by detecting individual motion patterns at the object level, with a specific emphasis on outdoor environments and perceptually degraded conditions. The effectiveness of the proposed framework is verified through rigorous experimentation, validating its capability to address

the identified challenges in autonomous navigation.

This chapter outlines a stepwise procedure for dynamic object detection in camera frames, employing a hybrid approach that combines machine learning and model-based solutions. The utilization of this hybrid method offers several advantages compared to end-to-end learning algorithms. Specifically, it allows for customization and adaptation to specific application constraints and requirements, facilitating the optimization of performance in diverse scenarios. In contrast, end-to-end learning methods may struggle to incorporate case-specific constraints effectively. Moreover, the hybrid method strikes a balance between data-driven estimation and the integration of prior knowledge. By leveraging geometrical reasoning, the framework enhances overall performance, complementing the capabilities of neural networks and enabling a more comprehensive understanding of the data processing task at hand.

4.1 Sensors

Detecting dynamic objects in 3D space from a single camera frame is challenging since the captured frames lack depth information. To overcome this loss of dimension, a pair of rectified stereo cameras with a fixed baseline is used. Since the cameras have a horizontal distance from each other, the pixel location of the same object will differ in the stereo images. This difference in pixel locations is called disparity, which is further explained in the stereo calibration section 4.3. By analyzing the disparities between the two stereo images, a depth map of the mutual field of view of the cameras can be reconstructed. This depth map provides useful information about the relative distances of the objects in the scene and enables the re-projection of features and objects from the image plane I to the 3D world frame W.

On the other hand, estimating the ego-vehicle motion and distinguishing it from the motion of objects in the environment is a challenging task, especially when the cameras themselves are in motion. In such cases, it is necessary to first estimate the ego-motion, which can then assist in segmenting the motion of surrounding objects. While motion estimation using cameras alone is pos-

sible, existing visual odometry methods often require substantial computation time and resources. To address this, an inertial measurement unit (IMU) is employed in this research to enhance robust ego-motion estimation. An IMU is a sensor consisting of three accelerometers and gyroscopes mounted on three orthogonal axes, enabling the measurement of linear acceleration and rotational rate around each axis. By integrating the data from the IMU sensors with the visual perception unit, more accurate ego-motion estimation can be achieved.

Moreover, the use of light detection and ranging sensors (LIDARs) is also discussed in this chapter as a ground truth generation sensor to verify the depth map obtained from stereo vision. LIDAR and camera data are also fused together to generate the final result of the dynamic object detection algorithm for comparison with the stereo vision method. Finally, to generate the ground truth data for the verification of localization results, a high-accuracy global navigation sensor system (GNSS) has been used, alongside real-time kinetic corrections. This sensor unit measures the latitude, longitude, and altitude of the vehicle which then is converted into a local universal transverse mercator (UTM) coordinate for convenience.

Before diving deep into the details of each module developed in this research, an overview of the complete process of this chapter is provided in a flowchart (Fig. 4.1). Inputs to this algorithm are raw stereo images and inertial measurements, and the output is a 2D detection of dynamic objects in the left image frame (${}^{c}I_{l}$).

4.2 Object Detection

The first step in the methodology of this chapter is detecting the objects of interest inside the visual frames from one of the cameras. It is to be noted that object detection is performed on the left camera only since the field of view of both cameras is similar. This will help with reducing the computational load of the algorithm by avoiding duplicate processing of visual data. For this chapter, a custom ROS implementation of the YOLOv4 [6] object

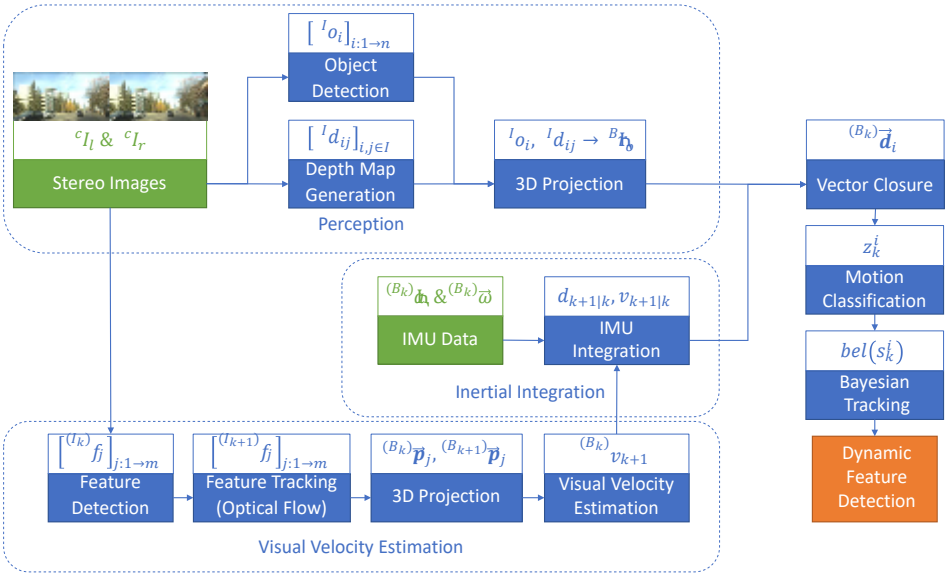


Figure 4.1: Dynamic object Detection flowchart. Green blocks are raw sensor input, blue blocks are data processing steps and the orange block is the output of the algorithm. (Brackets express lists)

detector is used to detect classes of objects on an outdoor scene that might be dynamic. These potentially dynamic object classes include cars, trucks, buses, pedestrians, cyclists, and motorbikes. Object detections will be represented with rectangular bounding boxes around the objects on the image frame (I). A sample of these bounding box detections including several data classes from the COCO dataset is represented in Fig. 4.2

The motion characteristics of potentially dynamic objects detected by YOLO are examined to classify their actual movement pattern. If these objects are confirmed to be dynamic, they are excluded from the localization process, leading to a significant improvement in localization accuracy. Conversely, if a potentially dynamic object is identified as stationary, it is included in localization. In this case, the features of the stationary object assist the visual perception unit in better comprehending the changes in the ego vehicle's position within the environment.

Figure 4.2: Bounding box detections for several classes in the COCO dataset. The potentially dynamic classes will be picked from all the detections based on their class number. These objects include cars, trucks, buses, pedestrians, cyclists, and motorbikes.

4.3 Stereo Camera Calibration

The next step involves the fusion of data acquired from the left and right cameras to reconstruct the missing dimension of the environment. To accomplish this, the initial step is to calibrate the cameras, mounted in a stereo configuration. This calibration process is done to measure the intrinsic parameters of the stereo cameras such as focal length and the optical center positions for each camera. These values are reported in a matrix known as the intrinsic camera matrix K which is provided in Eq. (4.1).

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

Moreover, the calibration also provides information about the relation between the pose of the cameras with respect to a known frame. This information is provided in an extrinsic matrix which includes a rotation matrix and a translation vector. The details of the extrinsic camera matrix are provided in Eq. (4.2).

$$[R|T] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{4.2}$$

The calibration is a meticulous procedure that involves several steps and requires substantial computational resources to accurately measure the intrinsic and extrinsic parameters as well as distortion characteristics. Initially, a large checkerboard with a known size (typically a 9×6 tile pattern) is placed in different poses in front of the stereo camera pair and used for pattern recognition. The frames captured by the cameras are used to localize the interior corners of the checkerboard, which serve as the reference pattern for parameter identification. By moving the checkerboard incrementally to cover all poses for a full excitation of patterns, a set of images are created with their corresponding corner detections and used for further computations. These post-processing steps include estimating the physical dimensions of the checkerboard pattern to establish the 3D world coordinates, matching the 2D corner coordinates between the cameras with their corresponding 3D world coordinates, optimizing the calibration by minimizing the reprojection error and estimating the intrinsic parameters (focal length, principal point, distortion coeficients) and extrinsic parameters (rotation and translation) for both cameras. This calibration process determines the geometric properties and distortion characteristics of the stereo camera system, which are essential for accurate stereo vision applications such as 3D reconstruction, depth estimation, and object tracking. An example set of the images used in the checkerboard stereo camera calibration is presented in Fig. 4.3.

Figure 4.3: Stereo camera intrinsic and extrinsic calibration using checkerboards.

Once the intrinsic and extrinsic matrices, as represented in Eq. (4.1) and Eq. (4.2), are obtained, the next step involves rectifying the image frames. Rectification is a crucial process that aims to transform the frames such that the image planes become virtually co-planar. When the image planes of the left

and right cameras are not co-planar, the search for pixels corresponding to the same points in the field of view becomes challenging. This can render disparity estimation either impossible or computationally unfeasible. By rectifying the image planes to be co-planar, the search space for matching pixels belonging to the same point in the environment is constrained to a horizontal line that passes through both images when they are displayed side by side. The co-planarity of the virtual image planes effectively shifts the epipolar center to infinity. Consequently, the lines that pass through the same object in both images become parallel with the y-axis of image coordinates. The rectification process is further illustrated in Fig. 4.4.

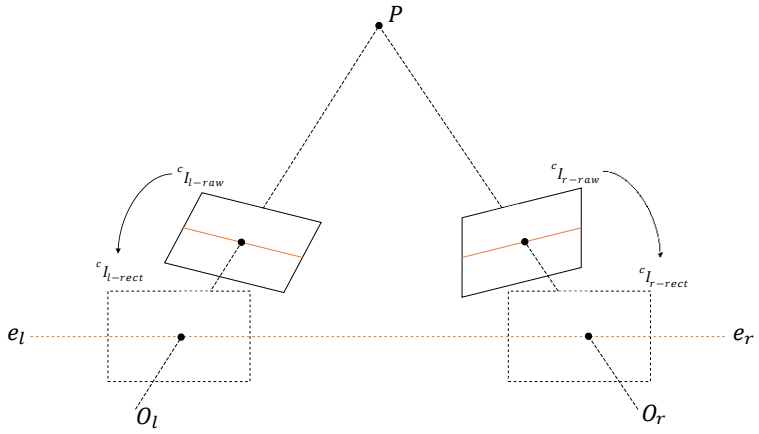


Figure 4.4: Epipolar geometry and rectification of image planes. Rectification creates virtual co-planar image planes to reduce the search space for similar pixels, to a unique horizontal line ($e_l e_r$) for eficient stereo matching performance.

4.4 Disparity Map Generation

Once the rectification and object detection steps have been performed in the 2D image space, the algorithm transitions to the 3D local space of the ego-vehicle. This transition enables the algorithm to perform 3D localization and motion estimation of both the ego-vehicle and the objects in the environment. However, one element is missing from the data gathered by the cameras and what has been perceived so far. This lost element is related to the fact that cameras capture the environment in 2D, lacking depth information for all pix-

els. Depth estimation plays a crucial role in this method as the final purpose is to study the independent motion of multiple objects in the 3D environment. While this task is relatively straightforward using stationary cameras, it becomes challenging in the case where cameras are mounted on a moving vehicle. This is due to the fact that differentiating between the motion of the ego-vehicle and the surrounding objects is dificult in 2D image space. Recovering depth can unlock a more comprehensive 3D space that can be used to describe the pose and motion characteristics of all objects more accurately. Moreover, by transitioning to 3D space, describing and segmenting the motion of surrounding objects is more realistic and does not require designing and testing exclusive 2D features incorporated into the images.

In order to perform motion classification by mapping 2D detections on the image plane to the 3D body frame, HITNET [106] is utilized to generate a dense disparity map that accurately estimates depth from the stereo image pair. HITNET leverages hierarchical iterative tiling and deep neural networks to provide a dense disparity map for all the pixels inside the left camera of the stereo setup. The resulting disparity map is a gray-scale image of the same size as the left camera image; a sample of this result is provided in Fig. 4.5. By having the disparity value of all pixels, a perspective transform can be employed to calculate the 3D location of each pixel in the body frame. This perspective transform utilizes the parameters estimated in the intrinsic calibration process discussed in Section 4.3.

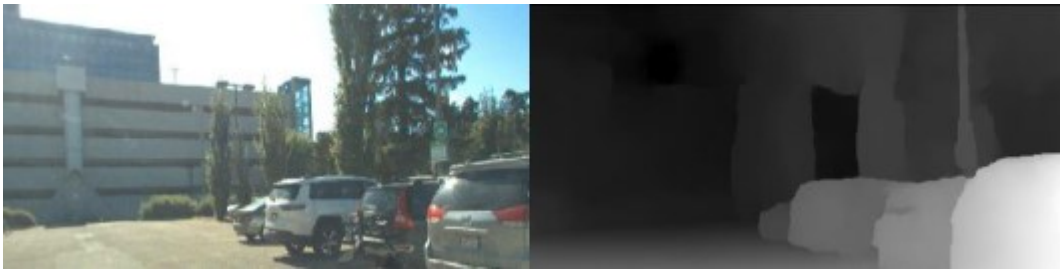


Figure 4.5: Resulting disparity map generated from stereo matching.

Projecting all the pixels with the perspective transform shown in Eq. (4.3) will be computationally expensive; thus only some pixels which represent the objects in the 2D images will be projected to the 3D body frame. Selected

pixels on the disparity map expressed in the image plane (I) are transformed into the body frame (B) using the perspective transformation matrix Q represented in Eq. (4.3). The transformation Q is solely dependent on camera parameters such as the left camera center coordinates (cx, cy), the focal length f, and the baseline of the stereo camera pair T. The resulting vector in the body frame represents the unscaled 3D components (X, Y, Z) which have to be divided by the scalar value of W to obtain the 3D coordinates of the object.

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}_B = \underbrace{\begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -\frac{1}{T} & 0 \end{bmatrix}}_{Q} \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix}_I \tag{4.3}$$

Utilizing this mathematical expression, it is possible to compute the relative location of individual objects or features within the images with respect to the body frame and utilize these values in the next sections of the analysis.

4.5 LIDAR as Ground Truth

Another way to localize the objects around the ego-vehicle is to use a Light Detection and Ranging sensor (LIDAR) to measure the depth of the environment directly. This method will generate precise position measurements for the objects surrounding the ego-vehicle, which are used as the localization ground truth for evaluating the visual node's performance. To achieve these precise measurements, the first step is to calibrate the left camera and the LIDAR together. This calibration provides the extrinsic parameters which will generate a homogeneous transformation between the left camera and the LIDAR (${}^{C}T_{L}$). By using this matrix, the 3D point clouds captured on the LIDAR frame (L) are first transformed into the camera frame (C). Thereafter, these points are projected onto the image plane (I), using the pinhole camera projection model as given in Eq. (4.4). Projected points that fall within the object bounding boxes discussed in Section 4.2 are identified and their indices are used to create multiple 3D regions of interest (ROI) in the LIDAR point cloud. These ROIs will be representing each object of interest that is detected on the image

frame. Visual filtering of 3D ROIs around objects of interest in the LIDAR point cloud eliminates the need for 3D object detection, significantly reducing the computational complexity.

$$\begin{bmatrix} x^i \\ y^i \\ 1 \end{bmatrix}_I = \begin{bmatrix} \frac{f}{c_{z^i}} & 0 & 0 & 0 \\ 0 & \frac{f}{c_{z^i}} & 0 & 0 \\ 0 & 0 & \frac{1}{c_{z^i}} & 0 \end{bmatrix} {}^{C}T_{L} \begin{bmatrix} x^i \\ y^i \\ z^i \\ 1 \end{bmatrix}_L \tag{4.4}$$

The 3D point cloud filtering process based on bounding box detections in the image frame may result in the inclusion of outliers that do not belong to the object of interest. These outliers arise from two common challenges encountered in outdoor environments. The first type of outliers is seen when occlusion takes place. For instance, if the object of interest, such as a car, is partially occluded by a tree, the LIDAR points reflected from the tree's surface may erroneously be included in the car's ROI. The second type of outlier includes LIDAR points located on distant surfaces that project close to the edges of 2D object bounding boxes. This is due to the fact that filtering a 3D space based on the re-projection of a rectangular bounding box on the image plane, creates a frustum of a pyramid (Fig. 4.6). This frustum which encloses the points inside the ROI could potentially include LIDAR points that are on extremely far objects. These outliers can be identified and filtered with Euclidean clustering which is discussed in detail in Section 4.6.

4.6 Clustering

Removing both types of outlier points from ROI point clouds is achieved through Euclidean clustering, as outlined in [112]. This method groups the points inside a point cloud based on their proximity to each other and has been used extensively in 3D object detection research as can be seen from [12], [40], [50], [51], [57]. By performing Euclidean clustering on individual ROI point clouds; multiple point clusters are generated which represent multiple neighborhoods of points. These clusters could be representing the object of interest or any other close-by surface; thus, cluster filtering is performed by comparing prior information about the geometry and the dimensions of the

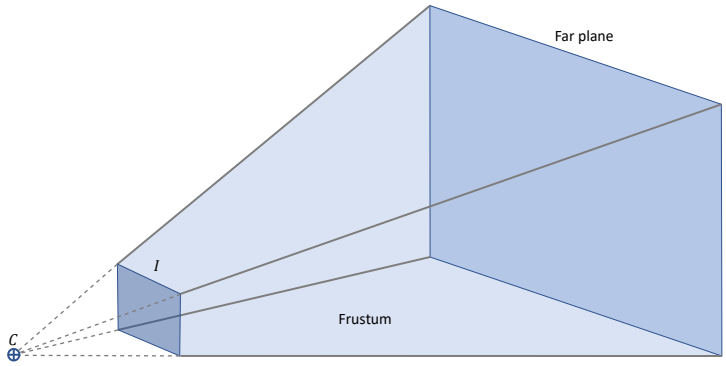


Figure 4.6: Visual frustum, created by filtering a 3D space based on the reprojection of a 2D bounding box on the image plane (I) onto a parallel plane which is far from the camera (C).

detected object with the measured aspect ratio and dimension of each cluster. This approach results in only one detection inside each ROI point cloud and concurrently rejects all of the clusters that are on other surfaces and objects in the ROI. Moreover, the frustum problem is also solved since the points creating the frustum are mostly far away in the scene and are present in the low-density areas of the ROI and Euclidean clustering disregards low-density neighborhoods in the point cloud.

After clustering the ROIs, the geometrical median of each filtered cluster is selected to present the center position of that object in the environment. This center position will be utilized as the ground truth for the visual 3D localization measurements of each object. These ground truth measurements are generated automatically through the proposed methodology in real-time. This eliminates the need for manual measurements from LIDAR data; making the evaluation process of the visual localization node more convenient. One significant bottleneck of this automatic ground truth generation algorithm is the 3D Euclidean clustering. However, by filtering the LIDAR data using visually-aware ROIs, the 3D search space is reduced significantly. This allows running multiple instances of Euclidean clustering for different ROIs, and real-time ground truth generation.

4.7 Ego-Vehicle Motion Estimation

Localizing objects seen in static camera frames and LIDAR point clouds in the local body frame has been well investigated in the literature. However, as the ego-vehicle is moving within the environment itself, each localization result will be in an exclusive frame that relates to the previous poses of the body frame with respect to a fixed world frame. As a result, estimating the motion of the ego-vehicle is crucial for calculating the transformations between the consecutive body frames. To achieve this, a GNSS sensor is used with real-time kinematic corrections from a base station to generate the ground truth for the ego-vehicle poses. This GNSS sensor fuses the data gathered from Global Positioning Satellites and the in-built IMU sensor to provide smooth acceleration, velocity, and positioning information. The accuracy of the ground truth for the ego vehicle's motion is less than 5cm which is adequate for the vehicle in an outdoor environment. For measuring the motion of the vehicle independent of the GNSS sensor, visual velocity estimation is done through the re-projection and tracking of 3D static features found on the left camera frame. The complete methodology is discussed further, later in this chapter.

4.8 Feature Tracking

The process of measuring the pose of an ego-vehicle from onboard cameras is called Visual Odometry VO. By analyzing the image frames captured from one or multiple onboard cameras, VO provides online incremental updates to the pose estimates of the vehicle as described in [10], [33]. However, visual odometry methods have long faced challenges related to computational complexity and varying lighting conditions [34], [122]. Another issue arises when tracking moving features, as it becomes impossible to distinguish between the motion of the feature and that of the ego-vehicle without additional information.

This chapter describes a novel velocity estimation for the ego-vehicle to aid with dynamic object detection through the camera frames. The first step is the extraction of features from the left image frame which is performed using a

Harris corner feature detector [36]. After extracting features, a filtration step is performed based on the object detection results. This is done to remove the features that are on potentially dynamic objects to avoid problems regarding the tracking of moving features. Based on the bounding box detections from YOLO (4.2), features that fall within the bounding boxes of potentially dynamic objects are removed from the tracking list. This will result in a more reliable tracking process and also reduces the number of features, reducing the computational load of the visual velocity estimation node.

Tracking static features is carried out using a simple and computationally eficient sparse optical-flow algorithm developed in [68]. Optical flow, also known as optic-flow, is referred to the study of patterns of apparent motion of objects and surfaces in the 2D image frame. These patterns are caused because of a relative motion between the observer camera and the different parts of the environment. The algorithm in [68] finds these flow patterns by constant motion assumption in a local neighborhood around a specific pixel under consideration. By forming the flow equations and solving them in the local neighborhood, a 2D flow vector is calculated which points from the query feature to its most probable new location in the new frame captured after the relative motion happens. Using this approach, all static features are tracked in 2D image frames and two sets of corresponding features are ready to be re-projected into the 3D body frame B for ego-motion analysis. The reprojection is done using the depth map created in Section 4.4, similar to the reprojection of dynamic objects from the image plane I to the body frame B explained in Eq. (4.3). Following this, both sets of tracked 2D features are reprojected on local body frame $B_{t=k}$ and $B_{t=k+1}$. Note that the body frames are not the same since the ego-vehicle is moving within the environment. To mea-sure this motion, a transformation is calculated between the two body frames $B_{t=k}$ and $B_{t=k+1}$ such that, the overall re-projection error of the feature set is minimized. To simplify the problem, the elevation of the features is dis-regarded, effectively reducing the search space from three dimensions to two. This assumption limits the transformation between the two body frames to three degrees of freedom instead of six. The three parameters considered are

the linear motion components along the x and y axes, and the yaw angle ψ around the z axis. By neglecting the elevation of features, the complexity of the problem is reduced while still capturing the essential aspects of the transformation between the frames. With this approach a simplified analysis of a planar ego-motion is carried out which aligns with the definition of the motion model explained in Section 4.9.

4.9 Motion Model

Defining a motion model for the ego-vehicle is an essential part of every autonomous driving system. Such a motion model serves as a foundational part of the algorithm, providing valuable information about the ego-motion of the autonomous agent. Without this information, the ability to detect dynamic objects only from moving camera frames becomes extremely challenging and computationally heavy. However, by leveraging a fully defined motion model, the performance of the perception unit can be enhanced for object detection and tracking. For the purpose of detecting moving objects, only a concise description of the ego-motion is suficient. This is mainly due to the fast data acquisition rate of sensors used in this project. The cameras and the inertial sensors used are operating at 20Hz which is relatively fast in comparison to the rate of change of the dynamics of vehicles moving in an outdoor setting. Moreover, controlling the ego-vehicle's motion is not the objective, thus reducing the need for a complex motion model describing all dynamic modes of the system in detail. Consequently, a simple two-degree of freedom (DOF) model is designed to describe the motion of the ego-vehicle in the environment. This model presumes the motion happens on an arc between every two consecutive frames (0.05s), which only requires a displacement vector ($\vec{d}_{k+1|k}$) and a change in the yaw angle ($\Delta\psi$) of the vehicle, to be fully defined. These two parameters are the most significant factors in the motion of the vehicle under normal driving conditions with minimal tire slip. This motion model is described in detail in Fig. 4.7.

Calculating the yaw angle of the vehicle is possible by integrating the an-

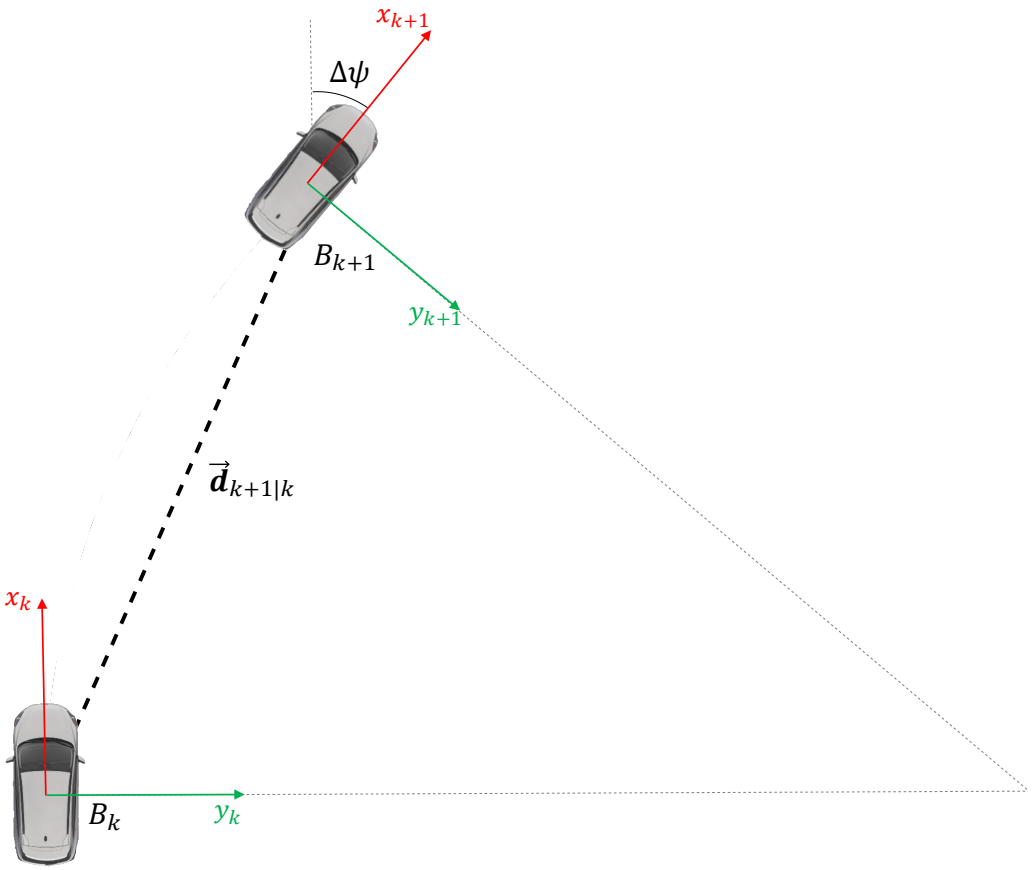


Figure 4.7: Definition of the 2DOF arc motion model of the ego-vehicle between two frames. Two degrees of freedom are represented as a linear motion with a magnitude of $|\vec{d}_{k+1|k}|$ and a change in yaw angle $\Delta\psi$.

gular velocity measurements coming from the gyroscope afixed to the body of the vehicle. However, the integration of noisy measurements will accumulate errors, and drift gradually from the actual yaw angle value. To avoid this, only the change in the yaw angle ($\Delta\psi$) is calculated by integrating once over new angular velocity measurements $\dot{\psi}_k$ around the z-axis of the body frame B_k. By focusing on the change in the yaw angle rather than its absolute value, the gradual integration drift effects on the estimation are mitigated. This is because this integration will only contain the sensor error at time t = k and does not suffer from any uncertainty accumulation over extended periods of time. Moreover, the absolute value of the orientation is not needed since all calculations happen relative to the body frame B_k.

In contrast, the estimation of the second parameter, $d_{k+1|k}$, in the motion

model presents greater challenges compared to estimating the change in yaw angle. This dificulty arises from the fact that IMU readings provide the second derivatives of linear motion, which necessitates the knowledge of initial conditions for double integration. Moreover, estimation drift can not be mitigated for double integration of uncertain measurements. Consequently, there is a need for a direct measurement of the initial condition, specifically the linear velocity of the vehicle. This is accomplished through the visual node by reprojecting the tracked features, as explained in Section 4.8. The detailed procedure for visual velocity estimation will be discussed in the subsequent section.

By establishing the motion modes of the vehicle, a homogeneous transformation can be formulated to map a body frame to the previous body frame. This transformation is constructed from a rotation matrix ${}^{(B_k)}R_{(B_{k+1})} \in R^{3\times3}$ and a translation vector ${}^{(B_k)}T_{(B_{k+1})} \in R^{3\times1}$. The rotation matrix aligns the two frames based on the yaw measurement $\Delta\psi$ and the translation vector shifts them to create the best overlap of the 3D features. The complete mathematical representation of this transformation is provided in Eq. (4.5).

$$ {}^{(B_k)}T_{(B_{k+1})} = \begin{bmatrix} \cos(\Delta\psi) & -\sin(\Delta\psi) & 0 & d\cos(\frac{\Delta\psi}{2}) \\ \sin(\Delta\psi) & \cos(\Delta\psi) & 0 & d\sin(\frac{\Delta\psi}{2}) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5} $$

The next section provides a description of velocity estimation based on static feature tracking and the definition of the motion model. This motion estimation is used to compensate for the relative motion of the scene with respect to the ego-vehicle and comprehend the absolute motion of each object in the environment.

4.10 Visual Velocity Estimation

After finding reliable static features on the left image and tracking them temporally, both sets are reprojected into their corresponding body frames. Since all features are selected on static surfaces and objects in the images, then their

reprojection should also be in the same place. Finding the best overlap of the features by calculating the optimal transformation between the two 3D sparse point clouds, can enable the algorithm to have a perception of the motion of the body frames. With this approach, ego-motion estimation becomes a simple optimization problem with only 2 degrees of freedom; one of which is the change in the yaw angle that is measured directly with the gyroscope as discussed in Section 4.9. The only other unknown parameter of this optimization is the magnitude of the linear ego-motion (d). By using the transformation matrix derived from the motion model Eq. (4.5) the reprojection error of each static feature (e_j) can be calculated using Eq. (4.6). It is to be noted that the magnitude of shift (d) is unknown and present in ${}^{(B_k)}T_{(B_{k+1})}$.

$$e_j = {}^{(B_k)}\vec{p}_j - {}^{(B_k)}T_{(B_{k+1})} \cdot {}^{(B_{k+1})}\vec{p}_j \tag{4.6}$$

As mentioned previously, the reprojection error component in the z-axis has no effect on the velocity estimation which is done in the xy plane. Consequently, by removing the 3rd dimension from Eq. (4.6), not only the accuracy of estimation is not affected, but also the computational complexity of the optimization process can be reduced. By removing the z dimension and reforming Eq. (4.6) such that the motion parameter (d) is isolated on one side Eq. (4.7) is created.

$$\begin{bmatrix} {}^{(B_k)}x \\ y \end{bmatrix}_j - \underbrace{\begin{bmatrix} \cos(\Delta\psi) & -\sin(\Delta\psi) \\ \sin(\Delta\psi) & \cos(\Delta\psi) \end{bmatrix}}_{r} \cdot \begin{bmatrix} {}^{(B_{k+1})}x \\ y \end{bmatrix}_j \approx \underbrace{\begin{bmatrix} \cos(\frac{\Delta\psi}{2}) \\ \sin(\frac{\Delta\psi}{2}) \end{bmatrix}}_{h} \times d_j \tag{4.7}$$

It should be noted that this equation is written with an ($\approx$) sign since the uncertainty of the sensors will result in different motion parameters (d_j) for each feature. Estimating a unique motion parameter which is closest to the actual value, can be dificult because of the high number of features and their different corresponding uncertainty levels. Thus, traditional optimization methods that form a cost function over all of the motion estimates for every feature, might not result in the optimal state estimate. To resolve this issue, a

recursive least square (RLS) equation is used to deal with sensor uncertainties and different errors for each feature, to obtain the best approximation of a single motion parameter for the ego-vehicle at each time stamp. First, Eq. (4.7) is used on all features to calculate a set of motion parameters. By finding the median of this set and choosing n features that have resulted in motion estimates closest to the median, we can formulate the RLS model as in Eq. (4.8).

$$\begin{bmatrix} x_1 \\ y_1 \\ \cdot \\ x_n \\ y_n \end{bmatrix}_{(B_k)} - R \begin{bmatrix} x_1 \\ y_1 \\ \cdot \\ x_n \\ y_n \end{bmatrix}_{(B_{k+1})} = H \begin{bmatrix} d_1 \\ d_2 \\ \cdot \\ d_n \end{bmatrix} + v_k \tag{4.8}$$

$$R = r_{2n\times 2n}, H = h_{2n\times n} \tag{4.9}$$

where, n is the number of chosen features that result in motion parameters closest to their median, r and h are defined in Eq. (4.7), and $v_k \in R_{n\times 1}$ represents the uncertainty of different features.

This model gets measurement input from n features which are extracted from the camera frame and reprojected to 3D using the generated depth map. The result will be $\frac{n}{2}$ motion parameter estimations which form a probability distribution function. These estimations and their corresponding covariance matrix, are initialized according to Eq. (4.10). The initial values for the estimates are all zeros, and the covariance matrix starts from a diagonal matrix with elements of 10. These values have been chosen based on the prior information about the magnitude and range of the motion parameters, which represent the displacement of the ego-vehicle in one-tenth of a second. The optimal motion parameter d_k at time t = k can vary based on the velocity of the vehicle, however, a starting magnitude of 0m and a covariance of $10m^2$ are good candidates. The covariance is set to start from a high value since at the initialization step, no information is available about the actual estimation.

However, choosing any higher value for the diagonal elements of the covariance matrix is not recommended as it may delay convergence.

$$\hat{D}_0 = O_{n\times 1}, P_0 = E[(d - \hat{d}_0)(d - \hat{d}_0)^T] \tag{4.10}$$

After proper initialization, the RLS updates the estimations by receiving the first measurement from the 3D features and the inertial sensor data, according to Eq. (4.11), (4.12) and (4.13).

$$K_k = P_{k-1}H_k^T(H_kP_{k-1}H_k^T + R_k)^{-1} \tag{4.11}$$

$$P_k = I - K_kH_kP_{k-1}(I - K_kH_k)^T + K_kR_kK_k^T \tag{4.12}$$

$$\hat{D}_k = \hat{D}_{k-1} + K_k(y_k - H_k\hat{X}_{k-1}) \tag{4.13}$$

First, an RLS gain is calculated based on the covariance of the previous step P_{k-1}, and the newly measured rotation block diagonal matrix R_k and measurement block diagonal matrix H_k. Following the calculation of RLS gain (K_k), the covariance matrix P_k and the estimation vector $\hat{D}_k$ are updated. By iteratively updating the estimated vector and covariance matrix based on new measurements, the recursive least squares (RLS) method provides a way to continuously improve the accuracy of the motion parameter estimates over time. This estimation method is particularly useful in our scenario where measurements are obtained sequentially and the uncertainties in the underlying system need to be estimated. The motion estimates gradually converge through multiple iterations of RLS and a vector of estimations forms around the actual value of the ego-motion parameter of the vehicle between two timestamps. To select a unique number based on the estimation vector, a histogram is formed and its center of area is picked as the number representing the motion of the ego-vehicle.

4.11 IMU Integration

The primary objective of this section is to achieve a consistent estimation of ego-motion, which can be computationally demanding when relying solely on the visual velocity estimation node. To overcome this challenge, a hybrid approach is employed where the visual velocity node measures velocity at predefined intervals, while the inertial measurement unit (IMU) predicts the velocity between these intervals. This approach improves computational eficiency and enhances robustness. The inclusion of information from the IMU compensates for limitations encountered by the visual node, such as challenging lighting conditions and feature tracking dificulties, thereby mitigating occasional failures in consistent velocity estimation. The synergistic utilization of data from both sources leads to more robust and reliable results.

IMU integration is done by assuming the motion of the vehicle has a constant acceleration between two timestamps. However, the value of the constant acceleration can change as new linear acceleration measurements are received. The integration is only carried out for the longitudinal motion of the vehicle. This is because the 2DOF motion model discussed in Section 4.9, only takes longitudinal and rotational motion modes into account. The formulations for the integration are presented in Eq. (4.14). Where $d_{k+1|k}$ is the longitudinal displacement from time $t = k$ to $t = k + 1$, a_k is the longitudinal acceleration reading from the previous timestamp measured by the IMU, v_k is the previously estimated velocity and finally t_s is the time passed between every two measurements.

$$d_{k+1|k} = a_k(\frac{t_s^2}{2}) + v_k t_s \tag{4.14}$$

The estimated velocity (v_k) comes from a single integration over the linear acceleration reading of the IMU. However, this process suffers from drift as mentioned previously. To eliminate this drift, the estimated velocity will be changed with visual velocity estimations at set intervals. This will keep the drift close to zero, making displacement estimations more accurate. The

following updates on the velocity of the ego-vehicle are calculated from Eq. (4.15).

$$v_k = v_{k-1} + a_{k-1} t_s \tag{4.15}$$

4.12 Vector Closure

After knowing the position of all objects of interest in two body frames B_k and B_{k+1}, and the ego-motion of the vehicle between t = k and t = k + 1, analysis of each object's motion can begin. Similar to 3D reprojected features, objects are also presented as single points in the 3D body frames. Consequently, each new 3D object position represented in B_{k+1}, can be transformed into the previous body frame B_k. This is done through the use of the same homogeneous transformation matrix presented in Eq. (4.5).

By performing this frame transformation, a predicted position ${}^{(B_k)}\vec{p}_i^{\,\prime}$ can be calculated for the i_{th} object according to Eq. (4.16). The predicted position of the i_{th} object in frame B_k is different than its measured position ${}^{(B_k)}\vec{p}_i$ at the previous time stamp. This allows for a comparison between predicted and previous measurements for the position of each object. Details of this transformation and projection are presented in Fig. 4.8. After this transformation is performed, all points are represented in a single body frame, the same as the case where the ego-vehicle is not moving.

$${}^{(B_k)}\vec{p}_i^{\,\prime} = {}^{(B_k)}T_{(B_{k+1})} \cdot {}^{(B_{k+1})}\vec{p}_i \tag{4.16}$$

Comparing the prediction and previous measurement of the position of the object i in frame B_k can provide a 3rd vector (${}^{(B_k)}\vec{d}_i$) which represents the motion of that object represented in the same frame.

4.13 Motion Classification

After the vector closure step, the motion characteristics of all objects of interest are determined and categorized as either static or dynamic. It is un-

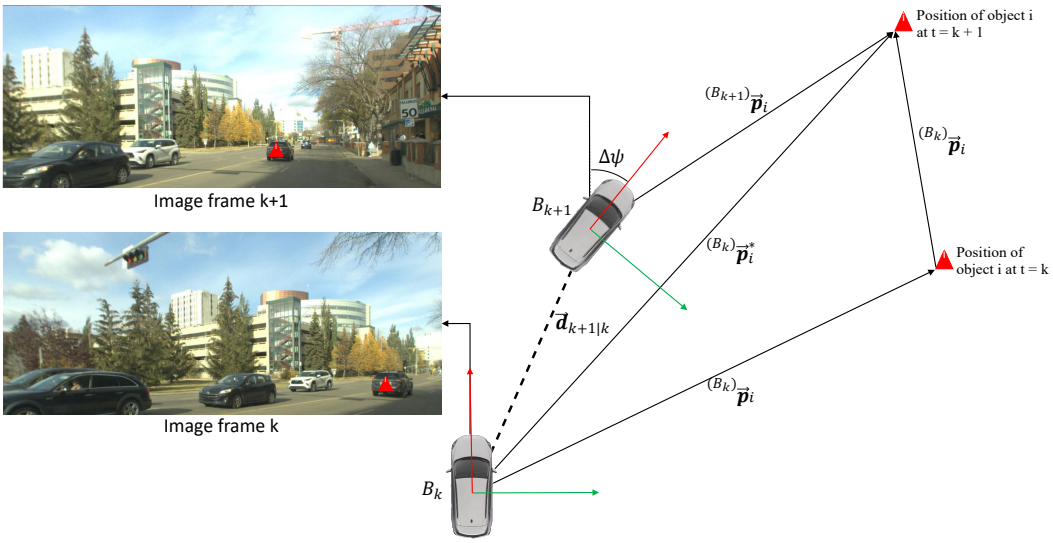


Figure 4.8: Overview of the reprojection of an object of interest from 2D image frames to the corresponding body frame at times t = k and k + 1. Between the initial body frame B_k and the two positions of the object i, a closure of vectors forms, enabling us to measure the motion of that object ($^{(B_k)}\vec{d}_i$). This vector is used to classify dynamic and static objects.

necessary to include small motions in the dynamic object list for two main reasons. Firstly, small motions do not result in significant changes in the position of features, which can be effectively compensated for through global and local minimization of errors in estimated ego-vehicle poses and reprojections. Therefore, including these small motions in the dynamic object list is unnecessary. Secondly, small motion detections can be attributed to sensor and process uncertainties, leading to dynamic classification for objects that are actually static. Disregarding small displacements helps to eliminate outliers from the final results and improves the overall accuracy.

On the other hand, the selection of an appropriate threshold value plays a crucial role in achieving optimal classification results. Fine-tuning this value allows for a balance between accuracy and consistency in the classifications. Increasing the threshold value disregards more small motion detections, leading to improved overall consistency. However, this may result in falsely detecting dynamic objects as static. Conversely, decreasing the threshold value provides a more accurate measure of object motion and can detect even subtle

displacements. However, this approach is not ideal as sensor uncertainties can introduce false detections for non-moving objects or objects with slow motion. Hence, finding the optimal motion threshold value is essential to achieve satisfactory classifications. This process is further discussed in Section 5.12.

4.14 Bayesian Tracking

To improve the temporal consistency of object motion classifications, a Bayesian filter is applied to track and analyze detection results. The binary static and dynamic states are transformed into a continuous probability domain between 0 and 1, enabling incremental updates to the detection probability. Subsequently, a Bayes filter is utilized for recursive updates on object states, with each object tracked using an ID generated by the 2D object detection algorithm described in Section 4.2. By applying Bayes' rule to the dynamic probability of the i^{th} object's state (s_k^i) at time t = k, given the prior measurements $z_{1:k}$ from t = 1 to t = k, the posterior probability of the state can be estimated. This is achieved by combining the prior probability and the likelihood of the measurements, as demonstrated in Eq. (4.17). It should be noted that the tracking process begins when the first measurement of a specific object is received. Thus the time reference (t = 1), varies for different objects, based on their respective entry times into the field of view of the cameras.

$$P(s_k^i \mid z_{1:k}^i) = P(z_k^i \mid s_k^i, z_{1:k-1}^i) P(s_k^i \mid z_{1:k-1}^i) \tag{4.17}$$

According to the Markov assumption, current measurements can be predicted based on the known state of the system at the same timestamp. This implies that the current measurement is independent of the prior measurements and is only correlated with the current state. Applying this Markov assumption to Eq. (4.17) will result in Eq. (4.18), as the prior measurements are not needed for predicting the probability of a measurement at the current time stamp.

$$P(s_k^i \mid z_{1:k}^i) = P(z_k^i \mid s_k^i) P(s_k^i \mid z_{1:k-1}^i) \tag{4.18}$$

Estimating the probability of the current state of the system at time $t = k$, solely based on all measurements until $t = k-1$ is challenging as no information is available about the most recent measurement (z_k^i). To address this, we can utilize the rule of total probability to leverage our knowledge about the previous belief of the system and estimate the probability of the current state based on the available measurement information. This approach allows us to derive a recursive formula for predicting the next belief of the system, taking into account the previous belief and the newly received measurements. The mathematical application of the rule of total probability is represented in Eq. (4.19).

$$P(s_k^i \mid z_{1:k}^i) = P(z_k^i \mid s_k^i) \int P(s_k^i \mid s_{k-1}^i, z_{1:k-1}^i) P(s_{k-1}^i \mid z_{1:k-1}^i) ds_{k-1}^i \tag{4.19}$$

Further analysis of Eq. (4.19), highlights another opportunity to use the Markov assumption to simplify the process. Specifically, the probability of the current state is independent of all the measurements prior to $t = k$. In other words, the probability of the current state of the system can be predicted only based on the previous state and is not affected by any measurements received before that. Applying this Markov assumption will result in Eq. (4.20)

$$P(s_k^i \mid z_{1:k}^i) = P(z_k^i \mid s_k^i) \int P(s_k^i \mid s_{k-1}^i) P(s_{k-1}^i \mid z_{1:k-1}^i) ds_{k-1}^i \tag{4.20}$$

By simplifying the process using the second Markov assumption, a recursive formula is obtained that can predict the current belief about the state of the system, based on the previous belief and the information gathered from the measurement and state transition models. This recursive Bayes filter formula is represented in Eq. (4.21).

$$bel(s_k^i) = \underbrace{P(z_k^i \mid s_k^i)}_{\text{measurement model}} \int \underbrace{P(s_k^i \mid s_{k-1}^i)}_{\text{state transition model}} bel(s_{k-1}^i) ds_{k-1}^i \tag{4.21}$$

The probability models, presented in the recursive Bayes formula are described as the following:

- Sensor model: Predicts the probability of a specific binary measurement given the current estimated state. It assigns a higher probability to detections that align with the predicted state. The probability is 95% for similar detections and 5% otherwise.
- State transition model: Predicts the probability of a new state occurring based on the previously estimated state. It assigns a higher probability to cases where the new state aligns with the previous state. The probability is 95% for consistent states and 5% for state changes.

Utilizing the sensor and state transition models, the recursive Bayes formula is completed and can be used for updating the state of each tracked object. For simplicity, the integral can be turned into a simple summation of two binary possibilities which is represented in Eq. (4.22). Where $\tilde{s}^j_{k-1}$ is the binary inverse of previous state of object j. The recursively calculated dynamic probability of each object can then be segmented into two halves using the central threshold probability of $\theta = 0.5$ to decide about the binary state of the object for visualization purposes.

$$bel(s^i_k) = P(z^i_k | s^i_k)[P(s^i_k | s^j_{k-1})bel(s^j_{k-1}) + P(s^i_k | \tilde{s}^j_{k-1})(1 - bel(s^j_{k-1}))] \quad (4.22)$$

By calculating the belief about the state of each object, we can decide about its dynamic/static class based on the dynamic probability model. This probability is then visualized in the shape of red (static) and green (dynamic) bounding boxes, on the left plane; This is furthermore discussed in the results section (5.7). Additionally, a detailed pseudo-code of the proposed method is provided in Algorithm 3.

4.15 Summary

In this chapter, by utilizing the semi-3D geometry of navigation environments, a novel visual-inertial dynamic object detection framework is presented to ad-

Algorithm 3: Dynamic object detection

Input : Raw images in the camera frames (${}^{c}I_{l}$ and ${}^{c}I_{r}$)
IMU measurements ${}^{(B_k)}\vec{a}$ and ${}^{(B_k)}\vec{\omega}$
camera parameters: c_x, c_y, f, T

Output: Object classification state list: $[bel(s_k^i)]$

while $k \geq 1$ do
  Extract the object bounding boxes using YOLO;
  Obtain centers of bounding boxes in both image frames: $[{}^{(I_k)}o_i]$ and $[{}^{(I_{k+1})}o_i]$, $i : 1 \rightarrow n$;
  Perform depth map generation using HITNET: ($[{}^{(I_k)}d_{ij}]$, $[{}^{(I_{k+1})}d_{ij}]$, $i, j \in I$);
  Re-project objects ${}^{(B_k)}\vec{r}_o$ and ${}^{(B_{k+1})}\vec{r}_o$ using Eq. (4.3);
  Estimate change in yaw using: $\Delta\psi_{(k:k+1)} = {}^{(B_k)}\vec{\omega}_{\psi} \times t_s$;
  Predict ego-vehicle displacement ($d_{k+1|k}$) from Eq. (4.14);
  ${}^{(B_k)}\vec{p}_i = {}^{(B_k)}T_{(B_{k+1})}.{}^{(B_{k+1})}\vec{p}_i$;
  Vector closure: ${}^{B_k}\vec{d}_i = {}^{(B_k)}\vec{p}_i - {}^{(B_{k+1})}\vec{p}_i$;
  $z_k^i = ||{}^{B_k}\vec{d}_i|| \geq Th_m$;
  Track object state with recursive Bayes formula Eq. (4.22);
end

dress perception challenges in autonomous navigation in highly dynamic environments and perceptually degraded conditions using onboard sensory units. The ego-motion of the vehicle was initially compensated by using a motion model over inertial data and estimating initial conditions based on measurements from stereo vision, effectively mitigating integration drift. Subsequently, displacements of visually detected objects obtained from a CNN network were analyzed using a semi-3D geometrical vector closure model. These displacements were then classified into static and dynamic classes using a carefully tuned threshold value, which facilitated the implementation of a stochas-tic filter with Bayesian tracking to enhance temporal consistency in motion classification. Additionally, point cloud clustering, disparity map generation, and consistent tracking were performed for both fixed- and moving-frame scenarios within the proposed framework. This hybrid model-based/data-driven approach exhibited numerous advantages over end-to-end learning solutions, including the incorporation of prior information for improved detection, adaptability to varying environmental conditions, generalization capability, and ro-

bustness to sensor uncertainties. The analysis yielded accurate classification of objects into dynamic and static classes, achieving a classification accuracy of approximately 90%. Consequently, major dynamic components in the vi-sual frames could be effectively filtered in real-time, resulting in frames that solely contain static parts that can seamlessly be integrated into online visual SLAM algorithms, which typically assume the prevalence of static features and landmarks in the environment, thereby facilitating autonomous navigation in highly dynamic environments.

Chapter 5

Experimental Results and Discussions

5.1 Infrastructure Aided Localization Results

The mobile robot used for experimental evaluation of the proposed localization framework is the Clearpath Jackal shown in Fig. 5.1. This robot is equipped with 4 wheels and a skid steering system. This mechanism allows the robot to control its motion by independently driving each wheel at a certain speed and direction for more agility. This control scheme facilitates the constant-acceleration motion model for the uncertainty-aware Kalman state observer with covariance adaptation. Further details of the Jackal robot are presented in table 5.1.

Table 5.1: Clearpath Jackal robot specifications.

Dimensions (m)			Linear motion		Rotational motion	
length	width	height	v_{max} (ms^{-1})	a_{max} (ms^{-2})	ω_{max} $(Rads^{-1})$	α_{max} $(Rads^{-2})$
0.508	0.430	0.250	2	4	20	25

Visual frames have been captured using an Intel Realsense T265 camera with a fisheye lens showcased in Fig. 5.1. Although this camera has two RGB sensors, only the frames from the right camera have been used for the proposed localization framework throughout this research. Also, Additional information from the internal IMU and generated stereo depth map are not used throughout this research. This is done to simulate the use of a low-cost

monocular camera sensor and prove its feasibility. Most important information about the Realsense T265 camera is provided in table 5.2.

Table 5.2: Intel Realsense T265 camera specifications.

Resolution (pixels)		sensor	
width	height	diagonal FOV°	Frames/second
848	800	173	30

Generating precise ground truth detections about the real location of the robot is crucial for the evaluation of the final results. Consequently, a Vicon motion capture camera system is used to generate the actual location of the robot in different scenarios. This motion capture camera system consists of multiple individual sensors installed all around the testing environment. Utilizing multiple high-resolution Vicon cameras from different angles creates the opportunity to detect Infrared (IR) markers attached to the robot with high accuracy.

Vicon cameras are calibrated with extreme precision prior to testing. Meticulous calibrations allow for a detection accuracy of around 1mm or even lower in our case. The location of the IR markers is then processed to provide meaningful full information about the pose of the robot; However, this is not the main focus of this research and thus we will not get much into the details of this process.

The coordinate frames showcased in Fig. 5.1, are denoted by {b} which is fixed to the robot body (located under the LiDAR); the camera frame {c} used for point cloud projection; {v} attached to the Vicon camera coordinate system, and {w} fixed to the location of a specific indoor feature, at the monitoring system unit. An initial calibration process has also been conducted between all coordinate system locations. This enables the automatic transformation of ground truth and localization measurements to the world frame.

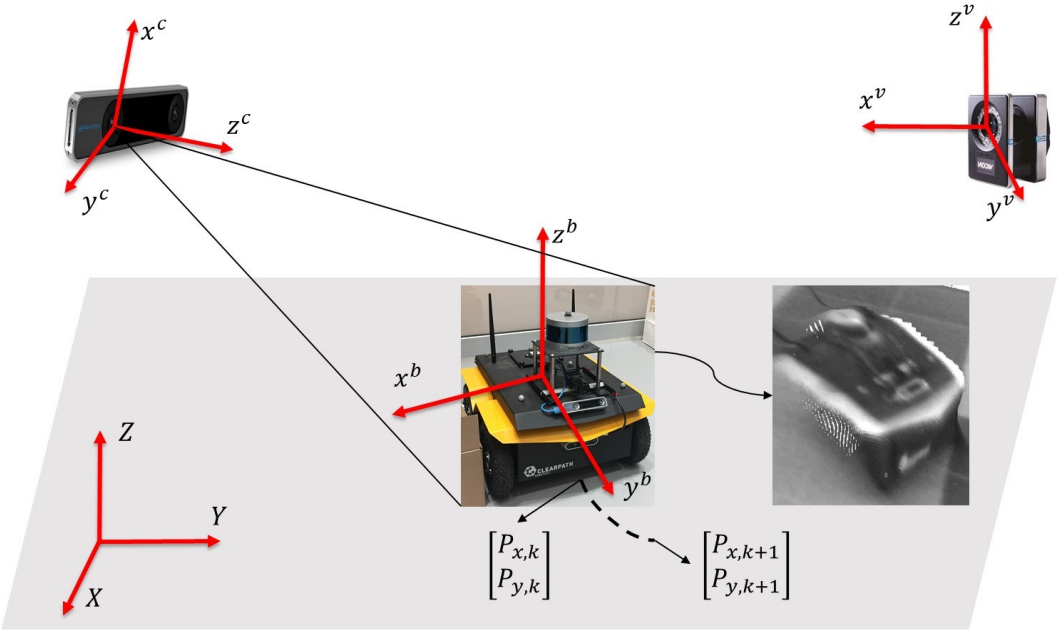


Figure 5.1: The experimental setup: the monocular vision and the Vicon system (as the ground truth) and a dense robot point cloud cluster.

5.2 Scenarios

To have a comprehensive evaluation of the results of the proposed algorithm, various trajectory tracking scenarios have been designed. Each scenario is designed to challenge the algorithm by simulating real-world problems that are encountered in the literature and industrial applications. The performance of the algorithm can be assessed by subjecting it to these scenarios. Moreover, the results obtained from these tests are used to measure the accuracy, precision, and robustness of this approach under different challenging conditions.

Designing the scenarios have been done to simulate two main categories of challenging conditions. The first category of scenarios is designed to address challenges associated with tracking highly complex trajectories. These trajectories include sharp turns, sudden changes in velocity, multiple changes in the direction of motion, and complex patterns that can challenge motion prediction models.

The second category is associated with problems and failure cases in the visual node. This involves simulating situations where detections are sporadic

due to multiple occlusions, or completely fail because of the limited field of view of the single camera used. By introducing such challenges, the robustness of the optimal state estimator is assessed. This category has been specifically designed for indoor environments where many humans are present and work alongside the robots in a symbiotic manner. This will be an often case in factories, warehouses, medical facilities, etc. after the industrial adoption of mobile robots.

5.3 Combined longitudinal/lateral trajectories

A broad range of robot motion in both longitudinal and lateral directions is covered in scenario 1 under the first category. The qualitative results are compared in Fig. 5.2 with pure visual-based detection and depth estimation. Each testing scenario is designed to measure a distinct ability of the proposed method. In scenario 1, although the range of longitudinal and lateral trajectories is greater than in other tests, the robot has been within the camera's field of view. Hence, visual detection is not disrupted. When depth estimation from the visual node drifts near the image frame boundaries, due to heavy blurring effects by undistortion, the uncertainty-aware observer handles such cases reliably through covariance adaptation, as demonstrated in Fig. 5.2.

As can be seen from Fig. 5.2, obviously, the motion model increases the accuracy of state estimation for trajectories far from the lens (i.e., lateral positions around 3.5m) where there is a small number of detection, and consequently inaccurate point cloud generation and depth estimation.

5.4 Temporary loss of image and occluded scenes

In scenario 2, the robot trajectory was intentionally set to go out of the undistorted frames (for almost 5s) to evaluate the robustness of the augmented framework to temporary loss of the robot coverage/detection in the image. The estimated robot states are provided in Fig. 5.3, where the capability of the proposed method in addressing scenarios with full loss of visual node information (due to long occlusion or moving out of coverage) is shown.

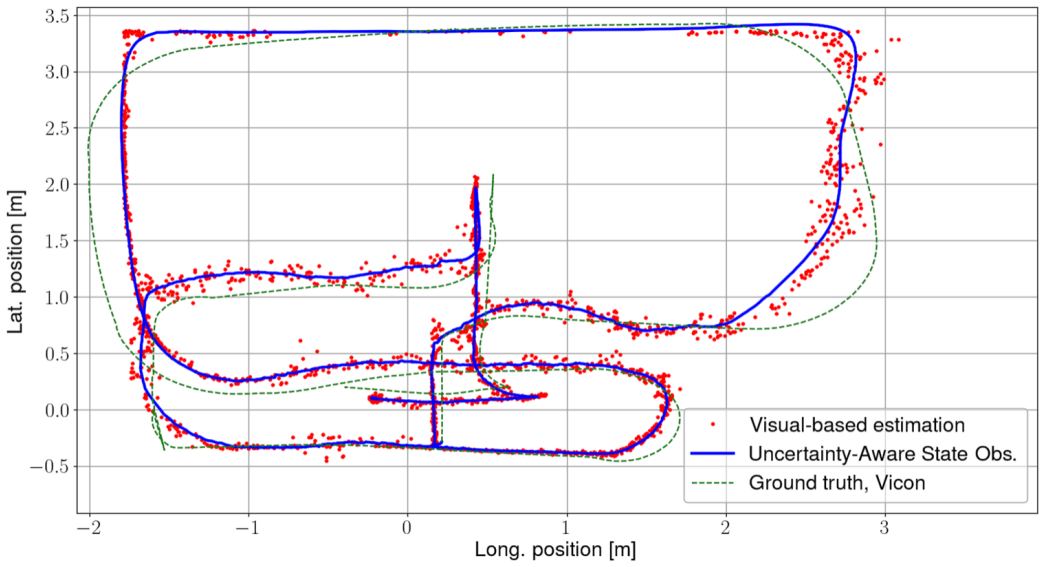


Figure 5.2: Robot trajectory and the estimation results for combined longitudinal and lateral maneuvers.

As can be seen from Fig. 5.3, as a halt in observations for a small portion of the robot trajectory, although no data is received from the visual node, the state observer can re-initiate correct estimation as soon as a new stream of visual measurements is received.

In the last four scenarios, the robot detection is interrupted multiple times due to occlusions that are common in indoor shared working environments (e.g., warehouses) where humans are present and work around autonomous mobile robots during robots' trajectory tracking operation. The estimated robot position for one of these occluded scenarios is shown in Fig. 5.4, where the method benefits from the uncertainty-aware state observer with acceleration prediction to address multiple loss of visual depth information.

Fig. 5.2-5.4 showcase the robustness of the developed framework under the aforementioned challenges by taking advantage of the motion model, ROI for point cloud generation, and learning-based acceleration prediction, where the Kalman observer reliance dynamically changes as a function of robot position to incorporate for sensor noise variances. Consequently, this adaptability results in reliable and consistent pose estimates in dynamic and human-robot-shared working environments even when the robot is partially visible.

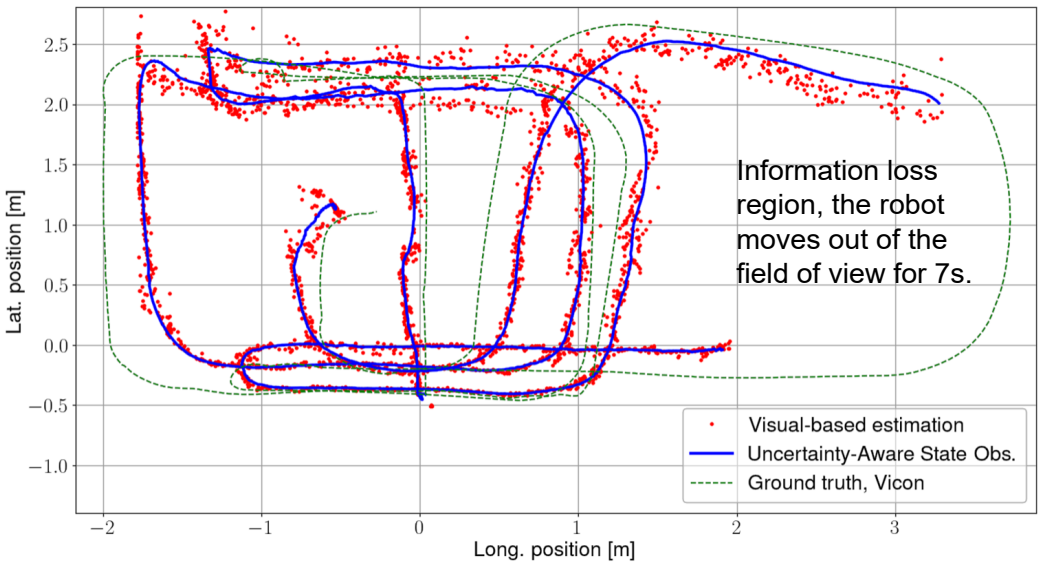


Figure 5.3: Visual-based estimated states augmented by the motion model and covariance adaptation during full loss of the robot visual tracking, due to trajectories beyond the camera's field of view.

Finding a region of interest via object detection prior to 3D localization decreases the chance of false positive clustering, and makes the algorithm capable of localizing the robot when it is partially visible from a camera perspective. Moreover, 3D detection is aided by taking advantage of the flat ground as the prior information for outlier removal. The performance of the detection for robot point cloud clustering during occlusion (by human or dynamic objects) and different camera orientations are demonstrated in Fig. 5.5 and 5.6, respectively, where the red point cluster shows the detected robot. In the aforementioned scenarios, the number of points in the environment point cloud has been uniformly down-sampled to reduce the computational burden.

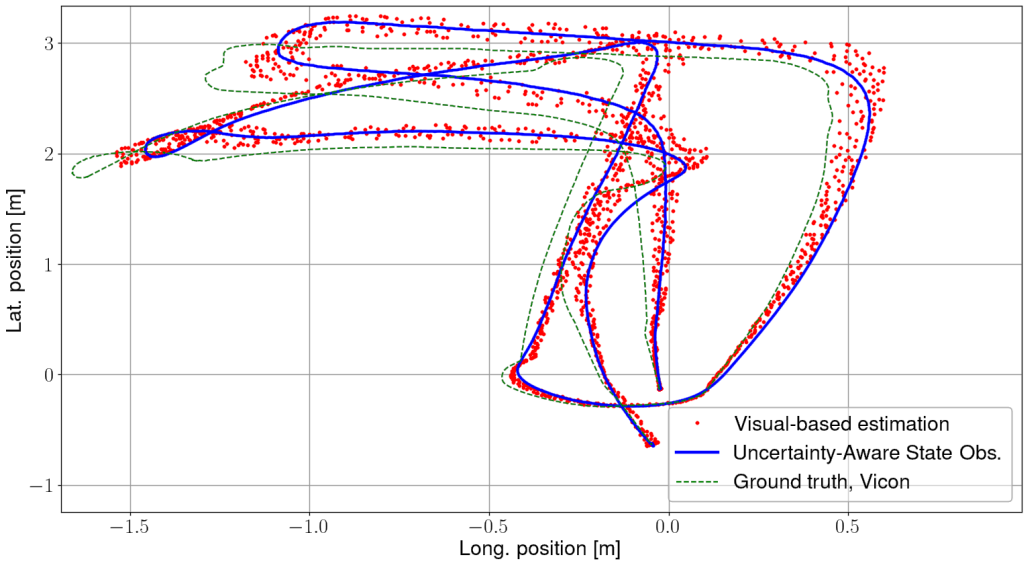


Figure 5.4: Estimated position of the robot with intermittent visual detection due to multiple occlusions with human presence.

5.5 Evaluation

In this section, position estimation error with respect to ground truth generated by Vicon cameras is analyzed in terms of root mean square error (RMSE) and average displacement error (ADE), to evaluate the performance of the proposed framework. All six testing scenarios mentioned previously, have been evaluated and the results are provided in table 5.3. The RMSE for scenario 2

Table 5.3: Statistical comparison of optimal estimates by the proposed method for various scenarios.

Scenario	Lat. RMSE (m)	Long. RMSE (m)	ADE (m)
1	0.3307	0.1412	0.2893
2	0.3645	0.2278	0.3768
3	0.1836	0.1928	0.2261
4	0.1993	0.1488	0.2269
5	0.2165	0.1927	0.2557
6	0.3296	0.1811	0.3390

is the largest (but still less than half of the robot length) due to long visual tracking loss (for almost 5s) when the robot moves out of the camera's field of view. Results can be improved significantly by the use of a camera with a

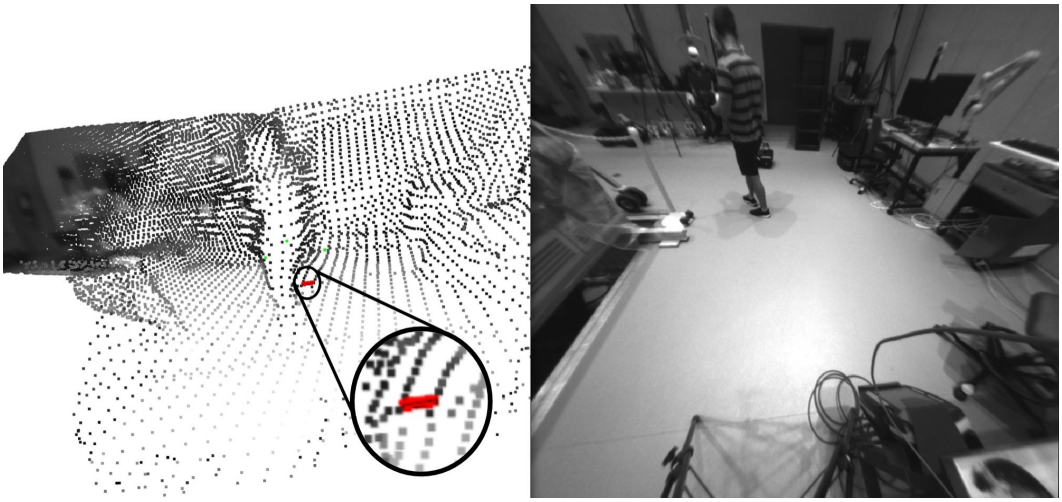


Figure 5.5: The robot point cloud clustering when partially visible due to human occlusion.

higher resolution which affects depth estimation and point cloud generation for robots operating at farther distances. The communication between the robot and the visual node (with the edge computing capability), which executes the uncertainty-aware state estimation and broadcasts only the pose information, will be conducted as the future work to enhance the accuracy of the localization for the robot's on-board motion planning and control systems.

Additionally, the trajectory of the robot has been studied in different segments consisting of close-range motion, far lateral and longitudinal motion, and cases where the robot is not consistently detected by the object detector due to occlusion. As shown in table 5.4, close-range localization is more precise which can be traced back to more reliable depth estimation for closer objects. In cases where the robot is moving farther from the camera, larger errors are observed, but no significant difference is seen between the longitudinal or lateral motion in the test scenarios.

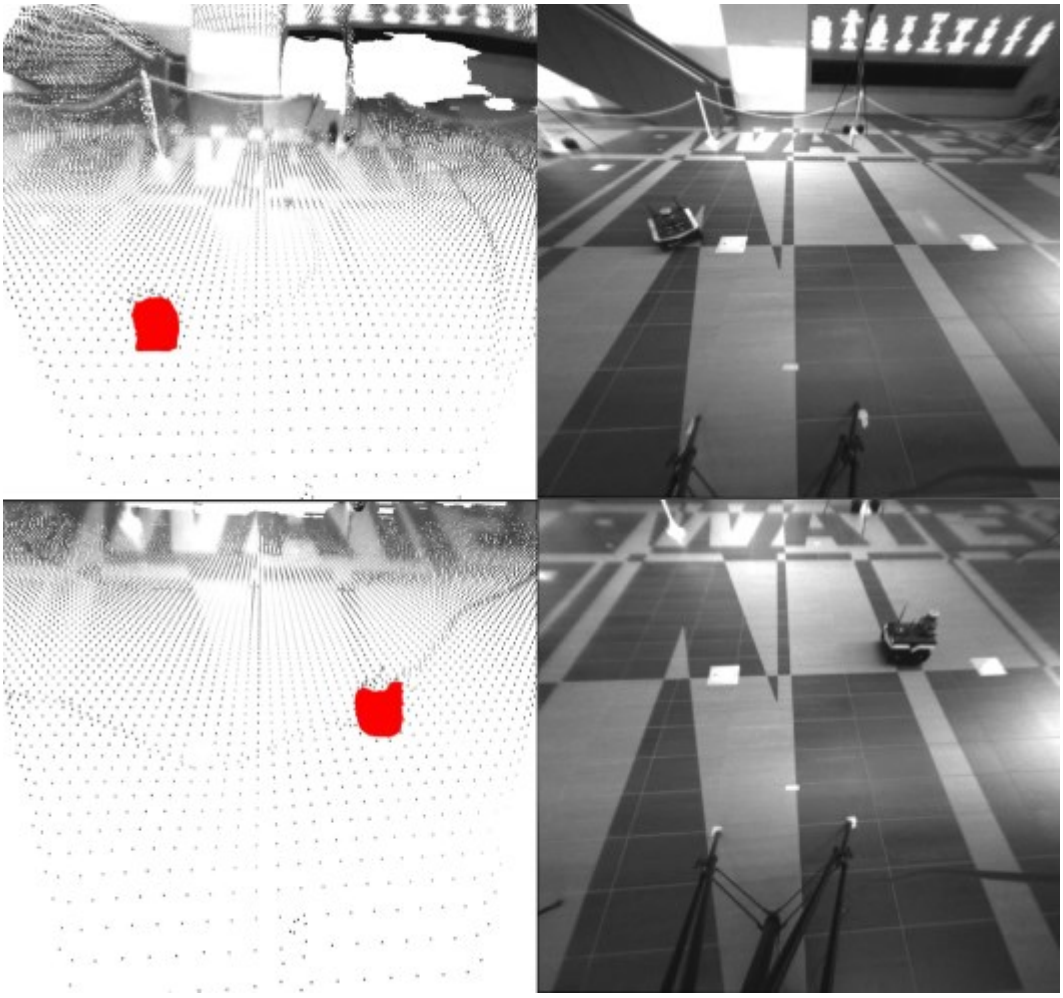


Figure 5.6: Robot point cloud clustering result with different camera orientations and robot positions.

5.6 Dynamic Object Detection Results

In this chapter, the experimental results obtained from the proposed dynamic object detection algorithm are presented. Prior to that, the technical details are elaborated upon, including the sensor types, specifications, and the coordinate frame convention employed in this research. Starting with the visual sensors, which serve as the primary source of information in this project, a stereo configuration is utilized, with two cameras positioned inside the ego-vehicle. These cameras are equipped with variable exposure times and have the capability to receive external signals for precise synchronization with the other sensors used on the vehicle. The specifications of the stereo cameras are

Table 5.4: Segmented statistical comparison

Scenario	Close range ADE (m)	Lat. motion ADE (m)	Long. motion ADE (m)	Occlusion ADE (m)
1	0.1582	0.2118	0.2980	
2	0.1656	0.2526	0.2194	
3	0.1344	0.2458	0.2484	
4	0.1484	0.2278	0.2351	0.2842
5	0.1684	0.2384	0.2421	0.2945
6	0.1427	0.2745	0.2946	0.3468

provided in Table 5.5 for reference.

Table 5.5: Stereo camera specifications.

$\text{Resolution}_{(\text{pixels})}$		sensor	
width	height	$\text{Focal length}_{(\text{pixels})}$	$\text{FPS}_{max(\text{Hz})}$
1600	1200	1325	60

Furthermore, to acquire ground truth information regarding the relative position of objects with respect to the vehicle's body frame, a LIDAR sen-sor has been installed on the vehicle's roof. This sensor is a mechanical 32-beam LIDAR that offers a panoramic view of the surroundings of the ego-vehicle. The high accuracy and 360-degree field of view of the LIDAR make it well-suited for obtaining reliable measurements of the environment's structure, which can be fused with camera data for acquiring ground truth information about the pose of any object around the ego-vehicle. Moreover, this sensor exhibits superior performance in adverse weather conditions compared to the cameras, ensuring the robustness of the ground truth data. This, in turn, en-ables a more comprehensive performance evaluation of the visual node within this project. Additional specifications of the LIDAR sensor can be found in Table 5.6.

On the other hand, generating accurate ground truth data for the ego-vehicle's position plays a crucial role in this thesis, as it facilitates a compre-hensive evaluation of the visual velocity estimation node employed to mitigate drift in IMU integration results. To accomplish this, a positioning sensor

Table 5.6: LIDAR specifications.

Detection		Field of view°		Sensor	
range(m)	accuracy(m)	horizontal	vertical	Resolution°	FPS(Hz)
1600	1200	360	40	0.2	10

is installed on the ego-vehicle to obtain real-time measurements of its location. This sensor leverages communications with the global navigation satellite system (GNSS) and real-time kinematic corrections to provide highly precise positioning data for the ego-vehicle. This location data is subsequently fused internally with IMU measurements to achieve more consistent and robust pose estimations, serving as the ground truth for the ego-vehicle's location. Additionally, the built-in 6 degrees of freedom (DOF) IMU of the GNSS sensor captures inertial measurements, including angular velocity and linear accelerations, which are utilized in the motion compensation node. A comprehensive overview of the technical specifications of this sensor can be found in Table 5.7.

Table 5.7: GNSS sensor specifications.

GNSS/INS Accuracy			IMU	Integrated sensor
position(m)	velocity(m/s)	heading°	DOF	FPS_{max}(Hz)
0.01	0.03	0.2	6	100

To ensure data synchronization, it is necessary for all sensors to record their measurements at a consistent refresh rate. In this setup, this is determined by the practical upper limit set by the 32-beam LIDAR. Therefore, all sensors are configured to record data at a rate of 10Hz, which is deemed suficient for autonomous navigation purposes. It should be noted that the final system has the capability to operate at higher frequencies, as ground truth data is not required during operation. Thus, the choice of refresh rate should be based on the specific requirements of any future projects and the available hardware resources. Furthermore, it is crucial to determine the relative positions of the

sensors with respect to the body frame to accurately fuse their data during the post-processing step. Consequently, all sensors are extrinsically calibrated and the resulting transformations are utilized throughout this project. An overview of the sensor frame positions, installed on the autonomous vehicle platform, is provided in Fig. 5.7.

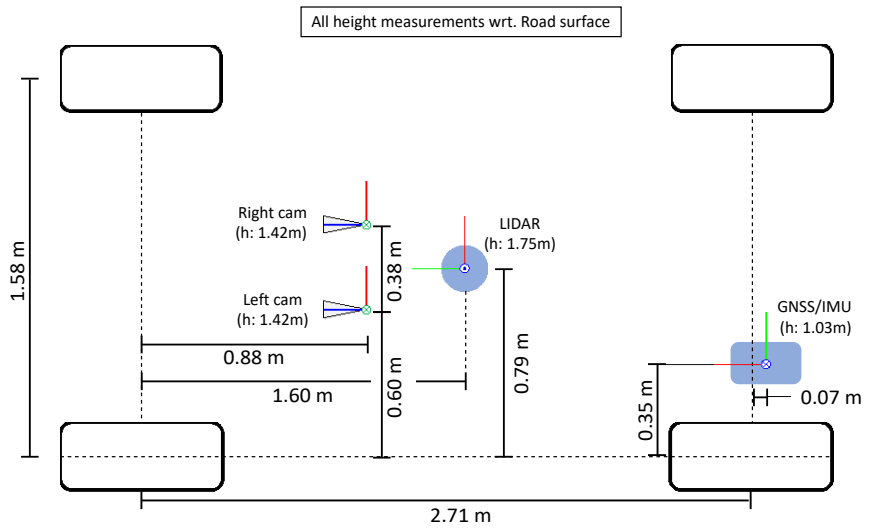


Figure 5.7: Frame diagram of all sensors installed on the autonomous vehicle platform, used for data acquisition.

5.7 Scenarios

In this section, a detailed discussion is provided regarding the experiments conducted in busy city streets, which served as an ideal example of highly dynamic environments, to evaluate the reliability, precision, and robustness of the algorithm. Meticulous consideration was given to designing scenarios that would challenge the algorithm in various ways, aiming to create a comprehensive evaluation process. Through this evaluation, both the weaknesses and strengths of the algorithm were revealed, offering valuable insights for future work and potential enhancements in the automated visual detection process.

Various challenges arise in busy city environments, resulting in noisy and uncertain measurements. For example, abrupt lighting changes can occur as the vehicle moves through the shadows of buildings, trees, and other vehicles in sunny conditions. These sudden lighting changes are frequent and inevitable in real-world scenarios, highlighting the need for a robust algorithm capable of handling such visual disturbances. Furthermore, the motion estimation and compensation process is complicated by the presence of multiple dynamic objects surrounding the vehicle. The assumption of stationary tracked features is challenged in highly dynamic environments, necessitating a higher level of perception to discern individual object motions. To comprehensively evaluate the performance of the proposed method, four distinct motion scenarios were considered for both the ego-vehicle and the objects in the environment. Each scenario was described in detail in the subsequent sections, followed by the presentation and discussion of the corresponding algorithm results.

5.8 Driving on Straight Busy Streets

The first scenario focuses on challenges encountered during straight driving, which is a significant driving condition. Demonstrating satisfactory performance in this scenario highlights the algorithm's effective handling of common challenges in detecting dynamic objects under typical driving conditions. The visual results for this scenario are depicted in Figure 5.8 for two different en-

vironments. Green and red bounding boxes represent detections of dynamic and static objects, respectively. The algorithm achieves accurate detection of dynamic objects through the utilization of novel visual-inertial motion estimation and Bayesian motion tracking techniques for surrounding objects. Additionally, the ego-motion of the vehicle is consistently and accurately estimated, demonstrating effective drift compensation based on static visual cues extracted from the images. The evaluation conducted in this scenario confirms the algorithm's capability to maintain correct state estimates over long drives without divergence or an increase in uncertainty.

Furthermore, the results demonstrate the accurate performance of the model-based state estimation used in the proposed framework under normal driving conditions in different environments. This functionality of the framework emphasizes the advantage of this research over fully data-driven approaches. Moreover, the accurate results shown in Figure 5.8 are achieved without the need for a large dataset or prolonged training time, aligning with the motivation behind choosing the model-based approach discussed in the previous motivation section (Section 1.1).

5.9 Intersections

The next scenario focuses on evaluating the algorithm's ability to accurately estimate orthogonal motion, which includes any motion that is not parallel to the course of the ego-vehicle. Objects moving perpendicular to the ego-vehicle tend to exit the camera's field of view more rapidly compared to objects moving parallel to the ego-vehicle. The limited field of view of the cameras imposes a time constraint on temporal object tracking and motion estimation. Many estimation models require multiple frames to converge to a correct estimated motion state. To address this challenge, the second scenario is designed to test the algorithm's detection capabilities at intersections, where orthogonal motion is commonly observed. The tests are conducted in two main categories, differing in terms of the ego-motion.

The first test is conducted when the vehicle is stationary behind a red

Figure 5.8: Straight driving scenario results. Each column of pictures represents a single driving test. The Left and right columns correspond to the highway and busy main street driving tests respectively.

light, observing the orthogonal motion of other vehicles from the first row of the trafic line. The second test evaluates the effect of visual motion estimation on these detections, where the vehicle is crossing an intersection. This test involves angled motion with respect to the ego-vehicle and multiple lane changes, creating a more challenging environment for detecting dynamic objects. Despite the presence of irregular or random motion patterns at intersections, the visual results shown in Figure 5.9 illustrate the accurate performance of the proposed method in detecting dynamic objects. This demonstrates the effectiveness of the geometrical vector closure model in estimating the motion of surrounding objects. Moreover, the Bayesian motion tracking technique rapidly converges to the correct motion state of the objects, as evidenced by

its accurate estimates after only the first frame. This rapid convergence is made possible through the innovative recursive Bayesian formula, in conjunction with the continuous dynamic probability model discussed in Section 4.14, which is used to track the binary states with continuous representation.

Figure 5.9: Results for scenarios at intersections. Each column of pictures represents a single driving test. The left column showcases the algorithm's capability to detect orthogonal motion with respect to the angle of view of the ego-vehicle. On the right column, the surrounding vehicles take a more random motion pattern which is angled with respect to the ego-vehicle's path.

5.10 Sharp Turns

The third scenario is designed to assess the framework's ability to detect the motion of other objects using onboard cameras while the ego-vehicle performs a sharp turn. This scenario presents several challenges. Firstly, the sharp

turn in the trajectory alters the relative motion patterns of surrounding objects from the ego-vehicle's perspective. Objects that were previously moving parallel to the ego-vehicle will now exhibit angular velocities relative to the observer. Additionally, the sharp turn introduces abrupt changes in sensor measurements, which can lead to incorrect or inconsistent object detections. Moreover, the rapid change in the field of view of the cameras during the turn can result in motion blur, posing a significant challenge for visual perception systems. Motion blur can lead to the loss or an insuficient number of features, which can impact the performance of the visual velocity estimation node. Consequently, the third scenario is designed to evaluate the framework's ability to handle challenges associated with sharp turn maneuvers.

The visual results shown in Figure 5.10 demonstrate that there is no significant degradation in the quality of motion segmentation. This can be attributed to the framework's design, which tracks the motion of objects rather than individual feature points. Object detection is a more robust form of perception compared to feature detection and matching. Furthermore, it is important to note that the framework estimates the ego-vehicle's motion by integrating gyro and accelerometer sensor measurements, enabling reliable inertial ego-motion estimation. This approach is specifically designed to mitigate the effects of unreliable visual motion estimations during sharp-turn scenarios, where motion blur is prevalent. The results also highlight the accurate performance of the hybrid ego-motion compensation module across different driving conditions.

5.11 Special Case Study

The final scenario aims to examine the impact of sudden changes in the trajectory of other vehicles, including turning maneuvers, acceleration, and deceleration. Special emphasis is placed on edge cases that occur at intersections in urban environments, as the accurate perception of the rapid motion changes of surrounding vehicles is crucial for the development of autonomous driving systems and road safety. Evaluating the performance of the proposed method in scenarios where various objects are moving in different directions and chang-

Figure 5.10: Sharp turn driving scenario results. The Left and right columns of pictures represent the frames from a left and right turn maneuver respectively.

ing their paths ensures the system's reliability, accuracy, and robustness in practical operation.

The results for this scenario consist of a single test drive comprising 10 frames, as illustrated in Figure 5.11. The evaluation begins with the ego-vehicle stopped behind a red light at an intersection, allowing the onboard cameras to observe multiple pedestrians and vehicles in motion. This demonstrates the algorithm's ability to detect the rapid changes in motion of the surrounding objects. In the second half of the test, the ego-vehicle initiates a sharp left turn maneuver behind another truck, introducing a challenge to the consistency of detections as both the ego-vehicle and the surrounding objects undergo state changes. However, the combination of a fast-responsive

Bayesian motion tracking algorithm and highly accurate motion estimations based on geometrical vector closure enables the framework to quickly adapt to these changes in the observed system.

Figure 5.11: Changing motion scenario results. The left column pictures the first half of the test where the ego-vehicle is stationary. However, the right column frames are captured during a left turn maneuver on the same test.

5.12 Quantitative Results

In this section, the performance of the proposed method is evaluated quantitatively based on the information obtained from the previously defined scenarios. To facilitate the numerical assessment, the binary detections are categorized into four groups, each with two attributes. These groups, as presented in Table 5.8, are described as follows:

- True Positive (TP): correct (True) detections of dynamic (Positive) objects. This indicates a successful detection of a dynamic object within the image frames and can provide a measure of detection accuracy.
- True Negative (TN): correct (True) detections of static (Negative) objects. This category also showcases the successful detection of a static object after data processing through the proposed method.
- False Positive (FP): incorrect (False) detections of dynamic (Positive) objects. This category refers to cases where a potentially dynamic object has been classified as static by mistake. These detections are identified based on their contradictions with the ground truth labels.
- False Negative (FN): incorrect (False) detections of static (Negative) objects. Indicating wrong dynamic detections for a static object.

Table 5.8: Category definitions used for quantifying dynamic object detection results

	Correct Detection	Wrong Detection
Dynamic Object	TP	FP
Static Object	TN	FN

The main intent of this research is to eliminate all dynamic objects from the visual frames to enhance the reliability of visual SLAM algorithms. Among the four categories of detections discussed earlier, false positive (FP) detections are considered the most detrimental. Mistakenly identifying dynamic objects as static can lead to unreliable frames that still contain dynamic regions and

compromises the accuracy and effectiveness of visual SLAM algorithms. To reduce the number of FP detection occurrences throughout experiments, an extensive hyper-parameter fine-tuning step has been conducted. By optimizing the parameters, the algorithm is structured with a bias towards classifying more objects as dynamic in cases where the object velocity is low and detections are uncertain. This will help to remove many FP cases as this bias will increase the number of dynamic detections.

On the other hand, this approach will create some FN detections because static objects may be falsely classified as dynamic. Nevertheless, this is not a major concern since falsely detecting a static object as dynamic will only result in removing some unnecessary regions from the images. Particularly, this will reduce the number of features that can be used for localization, but this reduction in static image regions is only by a small percentage since the majority of the frames will be still used for reliable feature detection and tracking purposes. Consequently, by introducing a dynamic detection bias in this algorithm, harmful FP detections are minimized and at the same time, reliable localization capabilities are maintained. Furthermore, by identifying high-probability regions in the camera frames that picture static parts of the environment, traditional visual SLAM algorithms can be used in highly dynamic environments without any concerns about the changes in the scene.

Precision criterion for the visual dynamic object detection results, is defined as the ratio of TP detections to total positive detections for all frames in a single test. This criterion measures the portion of dynamic objects that are identified correctly during a test drive. Another measure needed for the numerical evaluation of this algorithm is the recall criterion. Recall is defined as the ratio of TP detections to the total number of dynamic detections. This criterion can provide a numerical interpretation of the reliability of all dynamic detections. By combining precision and recall criteria, a metric can be defined as AP which measures the area under the precision-recall curve. This can provide a single numerical metric for evaluating the accuracy of the dynamic detections for a complete test drive. The mathematical equations regarding the numerical evaluation metrics are provided in Eq. (5.1) and (5.2).

$$Precision: P = \frac{TP}{TP + FP}, Recall: R = \frac{TP}{TP + FN} \tag{5.1}$$

$$AP = \frac{1}{n} \sum_{k=1}^{n} P_k R_k \tag{5.2}$$

For quantitative evaluations, manually labeled ground truth dynamic/static object segmentation datasets have been created for all of the testing scenarios provided in Section 5.8 to 5.11. All of the frames for each test drive are compared to the ground truth labels to calculate precision, recall and AP metrics, and complete results are provided in Table 5.9.

Table 5.9: Quantitative performance of dynamic object detection for all four scenarios.

Scenario	Average precision	Average recall	AP
Straight busy streets	0.967	0.912	0.893
Intersections	0.912	0.872	0.824
Sharp turns	0.923	0.895	0.861
Special case study	0.934	0.906	0.878

Chapter 6

Conclusions and Future Works

The primary objective of this thesis is the development of robust state estimation methods for autonomous navigation in highly dynamic environments and perceptually degraded conditions. This objective was studied from two perspectives: visual-based state estimation using stationary camera, and moving camera.

In this regard, two main approaches were proposed (and experimentally verified) to improve the accuracy and reliability of localization and perception of autonomous mobile agents in indoor and outdoor applications. The ini-tial part of this thesis centers around localizing a mobile robot moving within a dynamic indoor environment, utilizing an infrastructure-mounted camera. The robot was observed via a fixed monocular fisheye camera, and the noisy visual sensor measurements were fused with a constant acceleration motion model using an uncertainty-aware Kalman filter with covariance adaptation, helping reduce the overall uncertainty of the state estimation while using prior information about the motion of the robot over a horizon. By leveraging the visual information captured by the single low-cost camera, the proposed method showcased promising performance in terms of localization accuracy and robust trajectory estimation. The fusion of visual and motion information enabled the algorithm to mitigate measurement uncertainties and generate reliable trajectory estimates. A significant advantage of infrastructure-mounted sensors is their large and mostly unobstructed field of view to observe the operation of multiple robots in indoor/outdoor environments. This can decrease the cost

instead of installing onboard stereo cameras in each robot and increase the scale of operations through computing in the cloud. Moreover, having a centralized and fixed sensor unit will allow the use of more powerful hardware to increase the computational power, needed to manage networked robots.

Future avenues to improve the accuracy of the navigation solution using the proposed infrastructure-aided localization framework include: i) distributed state estimation (e.g., Kalman consensus filter) leveraging communication between the onboard estimator (which utilizes IMU, wheel encoders, and low-cost monocular camera) and the stationary sensor node mounted on infrastructure; and ii) developing constrained state observers and including semantic information for covariance adaptation. These two are recently under development at the NODE lab. This technique can also be used for autonomous driving applications (to address occlusion and enhance the Safety of Intended Functionality, SOTIF) through multiple stationary multimodal visual-LIDAR sensors for better coverage in highly dynamic environments.

The second objective of the thesis is to address the visual-based state estimation challenges in dynamic scenes for mobile robots/vehicles by designing robust dynamic object detection for reliable navigation using onboard visual and inertial sensory data. The proposed novel framework combines prediction over inertial data with the measurements from stereo vision-based state estimation to form a stochastic filter with Bayesian tracking for motion classification. The object classes that are detected for this work, range from pedestrians and cyclists to different types of vehicles such as cars, trucks, buses, etc. After detecting the potentially dynamic objects, their motion is furthermore investigated and classified into two static and dynamic groups. This is done by first estimating the ego-motion using a combination of integration over inertial measurements and visual cues gathered from static parts of the scene. This hybrid method allows for a more robust and accurate motion estimation and compensation process.

Experimental results in various dynamic scenes (e.g., driving in busy downtown areas) confirm a 90% accuracy on average for successfully detecting dynamic objects, while maintaining a very low computational load for real-time

performance required in autonomous driving, especially at high speed and at the tires capacity limit. By using this real-time algorithm, dynamic parts of the scene can be excluded from all localization processes, resulting in more accurate pose estimations for autonomous navigation. Moreover, this algorithm can be used for building static maps from the environment for indoor applications.

Potential future contributions could focus on i) optimizing computationally expensive algorithms involved in obtaining the visual dynamic regions. This optimization can enhance the eficiency of the process; and ii) incorporation of wheel odometry data within the general constrained semi-3D geometrical odometry as a motion model which is uncertain due to tires' longitudinal slip. The fusion of visual and wheel odometry data (through designing a new state observer) can lead to more precise estimations of the ego-motion, resulting in enhanced motion compensation results and lower uncertainties. However, the longitudinal slip will affect the accuracy of the state observer and needs to be considered as bounded uncertainty. This can be particularly valuable in applications where precise motion analysis and classification of dynamic objects are critical for navigation and decision-making tasks such as high-speed driving.

The outcomes of this research can considerably improve the navigational accuracy of autonomous mobile robots by designing robust and computationally-eficient state observers using available onboard sensory data and possible communication with stationary sensor nodes with wide real-time applications (with Edge computing capability) in intelligent transportation, and networked robots for services, delivery, and material handling in healthcare and manufacturing sectors.

References

[1] B. Anderson and J. B. Moore, "Detectability and stabilizability of time-varying discrete-time linear systems," SIAM Journal on Control and Optimization, vol. 19, no. 1, pp. 20–32, 1981.

[2] B. D. Anderson and J. B. Moore, Optimal filtering. Courier Corporation, 2012.

[3] I. Ballester, A. Fontan, J. Civera, K. H. Strobl, and R. Triebel, "Dot: Dynamic object tracking for visual slam," in 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2021, pp. 11 705–11 711.

[4] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," Lecture notes in computer science, vol. 3951, pp. 404–417, 2006.

[5] E. Binaghi, I. Gallo, G. Marino, and M. Raspanti, "Neural adaptive stereo matching," Pattern Recognition Letters, vol. 25, no. 15, pp. 1743–1758, 2004.

[6] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," CoRR, vol. abs/2004.10934, 2020. arXiv: 2004.10934. [Online]. Available: https://arxiv.org/abs/2004.10934.

[7] M. Brown, R. Szeliski, and S. Winder, "Multi-image matching using multi-scale oriented patches," in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), IEEE, vol. 1, 2005, pp. 510–517.

[8] M. Z. Brown, D. Burschka, and G. D. Hager, "Advances in computational stereo," IEEE transactions on pattern analysis and machine intelligence, vol. 25, no. 8, pp. 993–1008, 2003.

[9] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11, Springer, 2010, pp. 778–792.

[10] J. Campbell, R. Sukthankar, I. R. Nourbakhsh, and A. Pahwa, "A robust visual odometry and precipice detection system using consumer-grade monocular vision," Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 3421–3427, 2005.

[11] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: an accurate open-source library for visual, visual-inertial and multi-map SLAM," CoRR, vol. abs/2007.11898, 2020. arXiv: 2007.11898. [Online]. Available: https://arxiv.org/abs/2007.11898.

[12] B. Chen, H. Chen, D. Yuan, and L. Yu, "3d fast object detection based on discriminant images and dynamic distance threshold clustering," Sensors, vol. 20, no. 24, p. 7221, 2020.

[13] J. Chen and C. Yuan, "Convolutional neural network using multi-scale information for stereo matching cost computation," in 2016 IEEE International Conference on Image Processing (ICIP), IEEE, 2016, pp. 3424–3428.

[14] L. Chen, F. Rottensteiner, and C. Heipke, "Feature detection and description for image matching: From hand-crafted design to deep learning," Geo-spatial Information Science, vol. 24, no. 1, pp. 58–74, 2021. doi: 10.1080/10095020.2020.1843376. eprint: https://doi.org/10.1080/10095020.2020.1843376. [Online]. Available: https://doi.org/10.1080/10095020.2020.1843376.

[15] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul. 2017.

[16] H.-Y. Chung, C.-C. Hou, and Y.-S. Chen, "Indoor intelligent mobile robot localization using fuzzy compensation and Kalman filter to fuse the data of gyroscope and magnetometer," IEEE Transactions on industrial electronics, vol. 62, no. 10, pp. 6436–6447, 2015.

[17] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," Advances in neural information processing systems, vol. 29, 2016.

[18] W. Dai, Y. Zhang, P. Li, Z. Fang, and S. Scherer, "Rgb-d slam in dynamic environments using point correlations," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 1, pp. 373–389, 2020.

[19] K. K. Delibasis, V. P. Plagianakos, and I. Maglogiannis, "Real time indoor robot localization using a stationary fisheye camera," in Artificial Intelligence Applications and Innovations, H. Papadopoulos, A. S. Andreou, L. Iliadis, and I. Maglogiannis, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, isbn: 978-3-642-41142-7.

[20] L. Di Stefano, M. Marchionni, and S. Mattoccia, "A fast area-based stereo matching algorithm," Image and vision computing, vol. 22, no. 12, pp. 983–1005, 2004.

[21] L. Dreschler and H.-H. Nagel, "On the frame-to-frame correspondence between greyvalue characteristics in the images of moving objects," in GWAI-81: German Workshop on Artificial Intelligence Bad Honnef, January 26–31, 1981, Springer, 1981, pp. 18–29.

[22] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Eficient and robust 3d object recognition," in 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010, pp. 998–1005. doi: 10.1109/CVPR.2010.5540108.

[23] T. Eppenberger, G. Cesari, M. Dymczyk, R. Siegwart, and R. Dube, "Leveraging stereo-camera data for real-time dynamic obstacle detection and tracking," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2020, pp. 10528–10535.

[24] M. Everingham, L. Van Gool, and W. KI, "C., winn, j., & zisserman, a.(2010)," The PASCAL Visual Object Classes (VOC) Challenge, pp. 303–338,

[25] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," International journal of computer vision, vol. 88, pp. 303–338, 2010.

[26] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," Science Robotics, vol. 5, no. 40, eaaz9712, 2020.

[27] I. Fernandez, M. Mazo, J. L. Lazaro, et al., "Guidance of a mobile robot using an array of static cameras located in the environment," Autonomous Robots, vol. 23, no. 4, pp. 305–324, 2007.

[28] W. Forstner, Statistische Verfahren fur die automatische Bildanalyse und ihre Bewertung bei der Objekterkennung und-vermessung. Beck, 1991.

[29] R. Girshick, "Fast r-cnn," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.

[30] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.

[31] R. Gomez-Ojeda, F. A. Moreno, D. Scaramuzza, and J. G. Jimenez, "PL-SLAM: a stereo SLAM system through the combination of points and line segments," CoRR, vol. abs/1705.09479, 2017. arXiv: 1705.09479. [Online]. Available: http://arxiv.org/abs/1705.09479.

[32] A. Gonzalez, D. Vazquez, A. M. Lopez, and J. Amores, "On-board object detection: Multicue, multimodal, and multiview random forest of local experts," IEEE Transactions on Cybernetics, vol. 47, no. 11, pp. 3980–3990, 2017. doi: 10.1109/TCYB.2016.2593940.

[33] R. Gonzalez, F. Rodriguez, J. Guzman, C. Pradalier, and R. Siegwart, "Combined visual odometry and visual compass for off-road mobile robots localization," Robotica, vol. 30, pp. 1–14, Oct. 2012. doi: 10.1017/S026357471100110X.

[34] R. Gonzalez, F. Rodriguez, J. Guzman, C. Pradalier, and R. Siegwart, "Control of off-road mobile robots using visual odometry and slip compensation," Advanced Robotics, vol. 27, Aug. 2013. doi: 10.1080/01691864.2013.791742.

[35] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, IEEE, vol. 2, 2005, pp. 1458–1465.

[36] C. Harris and M. Stephens, "A combined corner and edge detector," in Proceedings of the 4th Alvey Vision Conference, 1988, pp. 147–151.

[37] C. Harris, M. Stephens, et al., "A combined corner and edge detector," in Alvey vision conference, Citeseer, vol. 15, 1988, pp. 10–5244.

[38] K. He, G. Gkioxari, P. Dollar, and R. B. Girshick, "Mask R-CNN," CoRR, vol. abs/1703.06870, 2017. arXiv: 1703.06870. [Online]. Available: http://arxiv.org/abs/1703.06870.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[40] D. Held, D. Guillory, B. Rebsamen, S. Thrun, and S. Savarese, "A probabilistic framework for real-time 3d segmentation using spatial, temporal, and semantic cues.," in Robotics: Science and Systems, vol. 12, 2016.

[41] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 2, pp. 328–341, 2008. doi: 10.1109/TPAMI.2007.1166.

[42] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. arXiv: 1502.03167 [cs.LG].

[43] S. J. Julier, "The scaled unscented transformation," in Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301), IEEE, vol. 6, 2002, pp. 4555–4559.

[44] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," Proceedings of the IEEE, vol. 92, no. 3, pp. 401–422, 2004.

[45] J. Jung, S.-M. Lee, and H. Myung, "Indoor mobile robot localization and mapping based on ambient magnetic fields and aiding radio sources," IEEE Transactions on Instrumentation and Measurement, vol. 64, no. 7, pp. 1922–1934, 2015.

[46] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[47] R. Kang, L. Xiong, M. Xu, J. Zhao, and P. Zhang, "Vins-vehicle: A tightly-coupled vehicle dynamics extension to visual-inertial state estimator," in 2019 IEEE Intelligent Transportation Systems Conference (ITSC), IEEE, 2019, pp. 3593–3600.

[48] J. Kannala and S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," IEEE transactions on pattern analysis and machine intelligence, vol. 28, pp. 1335–40, Sep. 2006. doi: 10.1109/TPAMI.2006.153.

[49] A. Kendall, H. Martirosyan, S. Dasgupta, et al., "End-to-end learning of geometry and context for deep stereo regression," in Proceedings of the IEEE international conference on computer vision, 2017, pp. 66–75.

[50] K. Klasing, D. Wollherr, and M. Buss, "A clustering method for eficient segmentation of 3d laser data," in 2008 IEEE international conference on robotics and automation, IEEE, 2008, pp. 4043–4048.

[51] K. Klasing, D. Wollherr, and M. Buss, "Realtime segmentation of range data using continuous nearest neighbors," in 2009 IEEE International Conference on Robotics and Automation, IEEE, 2009, pp. 2431–2436.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, 2017.

[53] A. Kuznetsova, H. Rom, N. Alldrin, et al., "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," International Journal of Computer Vision, vol. 128, no. 7, pp. 1956–1981, 2020.

[54] T. Lefebvre*, H. Bruyninckx, and J. De Schutter, "Kalman filters for non-linear systems: A comparison of performance," International journal of Control, vol. 77, no. 7, pp. 639–653, 2004.

[55] R. Lenain, B. Thuilot, C. Cariou, and P. Martinet, "Mixed kinematic and dynamic sideslip angle observer for accurate control of fast off-road mobile robots," Journal of Field Robotics, vol. 27, no. 2, pp. 181–196, 2010.

[56] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in 2011 International conference on computer vision, Ieee, 2011, pp. 2548–2555.

[57] H. Li, Z. Wang, G. Yu, et al., "3dsg: A 3d lidar-based object detection method for autonomous mining trucks fusing semantic and geometric features," Applied Sciences, vol. 12, no. 23, p. 12 444, 2022.

[58] X. Liang, H. Wang, Y.-H. Liu, W. Chen, and Z. Jing, "Image-based position control of mobile robots with a completely unknown fixed camera," IEEE Transactions on Automatic Control, vol. 63, no. 9, pp. 3016–3023, 2018. doi: 10.1109/TAC.2018.2793458.

[59] J. Lin, H. Zhu, and J. Alonso-Mora, "Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments," in 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2020, pp. 2682–2688.

[60] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2117–2125.

[61] T.-Y. Lin, M. Maire, S. Belongie, et al., "Microsoft coco: Common objects in context," in Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13, Springer, 2014, pp. 740–755.

[62] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection," in Proceedings CVPR IEEE computer society conference on computer vision and pattern recognition, IEEE, 1996, pp. 465–470.

[63] W. Liu, D. Anguelov, D. Erhan, et al., "Ssd: Single shot multibox detector," in Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14, Springer, 2016, pp. 21–37.

[64] X. Liu, G. V. Nardari, F. C. Ojeda, et al., "Large-scale autonomous flight with real-time semantic slam under dense forest canopy," IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 5512–5519, 2022.

[65] S. Y. Loo, S. Mashohor, S. H. Tang, and H. Zhang, "Deeprelativefusion: Dense monocular slam using single-image relative depth prediction," in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 6641–6648. doi: 10.1109/IROS51168.2021.9636504.

[66] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International journal of computer vision, vol. 60, pp. 91–110, 2004.

[67] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in IJCAI'81: 7th international joint conference on Artificial intelligence, vol. 2, 1981, pp. 674–679.

[68] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (darpa)," pp. 121–130, Apr. 1981.

[69] Q. Luo, J. Zhou, S. Yu, and D. Xiao, "Stereo matching and occlusion detection with integrity and illusion sensitivity," Pattern recognition letters, vol. 24, no. 9-10, pp. 1143–1149, 2003.

[70] M. H. Mamduhi, E. Hashemi, J. S. Baras, and K. H. Johansson, "Event-triggered add-on safety for connected and automated vehicles using road-side network infrastructure," IFAC-PapersOnLine, vol. 53, no. 2, pp. 15 154–15 160, 2020.

[71] N. Mayer, E. Ilg, P. Hausser, et al., "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 4040–4048.

[72] K. Mikolajczyk and C. Schmid, "Scale & afine invariant interest point detectors," International journal of computer vision, vol. 60, pp. 63–86, 2004.

[73] A. Moffatt, E. Platt, B. Mondragon, A. Kwok, D. Uryeu, and S. Bhandari, "Obstacle detection and avoidance system for small uavs using a lidar," in 2020 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2020, pp. 633–640.

[74] E. Mohammadbagher, N. P. Bhatt, E. Hashemi, B. Fidan, and A. Khajepour, "Real-time pedestrian localization and state estimation using moving horizon estimation," in 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), 2020, pp. 1–7. doi: 10.1109/ITSC45102.2020.9294306.

[75] H. P. Moravec, "Visual mapping by a robot rover," in Proceedings of the 6th International Joint Conference on Artificial Intelligence - Volume 1, ser. IJCAI'79, Tokyo, Japan: Morgan Kaufmann Publishers Inc., 1979, pp. 598–600, isbn: 0934613478.

[76] K. Muhlmann, D. Maier, J. Hesser, and R. Manner, "Calculating dense disparity maps from color stereo images, an eficient implementation," International Journal of Computer Vision, vol. 47, pp. 79–88, 2002.

[77] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147–1163, 2015. doi: 10.1109/TRO.2015.2463671.

[78] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," CoRR, vol. abs/1610.06475, 2016. arXiv: 1610.06475. [Online]. Available: http://arxiv.org/abs/1610.06475.

[79] L. Nalpantidis, G. C. Sirakoulis, and A. Gasteratos, "Review of stereo matching algorithms for 3d vision," in 16th International Symposium on Measurement and Control in Robotics 21-23 June 2007-Warsaw, POLAND, 2007.

[80] H. Oleynikova, D. Honegger, and M. Pollefeys, "Reactive avoidance using embedded stereo vision for mav flight," in 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015, pp. 50–56.

[81] A. A. Ortega Jimenez and J. Andrade-Cetto, "Segmentation of dynamic objects from laser data," in ECMR 5th European Conference on Mobile Robots, 2011, pp. 115–121.

[82] C. Premebida, J. Carreira, J. Batista, and U. Nunes, "Pedestrian detection combining rgb and dense lidar data," in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 4112–4117. doi: 10.1109/IROS.2014.6943141.

[83] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Oct. 2019.

[84] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004–1020, 2018. doi: 10.1109/TRO.2018.2853729.

[85] C. Ramer, J. Sessner, M. Scholz, X. Zhang, and J. Franke, "Fusing low-cost sensor data for localization and mapping of automated guided vehicle fleets in indoor applications," in 2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), 2015, pp. 65–70. doi: 10.1109/MFI.2015.7295747.

[86] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2020.

[87] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[88] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement. 2018," arXiv preprint arXiv:1804.02767, vol. 20, 1804.

[89] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271.

[90] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," Advances in neural information processing systems, vol. 28, 2015.

[91] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An eficient alternative to sift or surf," in 2011 International conference on computer vision, Ieee, 2011, pp. 2564–2571.

[92] O. Russakovsky, J. Deng, H. Su, et al., "Imagenet large scale visual recognition challenge," International journal of computer vision, vol. 115, pp. 211–252, 2015.

[93] A. Saha, B. C. Dhara, S. Umer, K. Yurii, J. M. Alanazi, and A. A. AlZubi, "Eficient obstacle detection and tracking using rgb-d sensor data in dynamic environments for robotic applications," Sensors, vol. 22, no. 17, p. 6537, 2022.

[94] Y. Sawaragi and T. Katayama, "Performance loss and design method of kalman filters for discrete-time linear systems with uncertainties," International Journal of Control, vol. 12, no. 1, pp. 163–172, 1970.

[95] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," International journal of computer vision, vol. 47, pp. 7–42, 2002.

[96] M. Shakeri, S. Y. Loo, H. Zhang, and K. Hu, "Polarimetric monocular dense mapping using relative deep depth prior," IEEE Robotics and Automation Letters, vol. 6, no. 3, pp. 4512–4519, 2021. doi: 10.1109/LRA.2021.3068669.

[97] W. Shi, J. Li, Y. Liu, D. Zhu, D. Yang, and X. Zhang, "Dynamic obstacles rejection for 3d map simultaneous updating," IEEE Access, vol. 6, pp. 37 715–37 724, 2018.

[98] M. Siam, H. Mahgoub, M. Zahran, S. Yogamani, M. Jagersand, and A. El-Sallab, "Modnet: Moving object detection network with motion and appearance for autonomous driving," arXiv preprint arXiv:1709.04821, 2017.

[99] M. Simon, S. Milz, K. Amende, and H.-M. Gross, "Complex-yolo: Real-time 3d object detection on point clouds," arXiv preprint arXiv:1803.06199, 2018.

[100] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[101] X. Song, X. Zhao, H. Hu, and L. Fang, "Edgestereo: A context integrated residual pyramid network for stereo matching," in Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part V 14, Springer, 2019, pp. 20–35.

[102] J. Sun, N.-N. Zheng, and H.-Y. Shum, "Stereo matching using belief propagation," IEEE Transactions on pattern analysis and machine intelligence, vol. 25, no. 7, pp. 787–800, 2003.

[103] J. Sun, N.-N. Zheng, and H.-Y. Shum, "Stereo matching using belief propagation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 7, pp. 787–800, 2003. doi: 10.1109/TPAMI.2003.1206509.

[104] Y. Sun, M. Liu, and M. Q.-H. Meng, "Improving rgb-d slam in dynamic environments: A motion removal approach," Robotics and Autonomous Systems, vol. 89, pp. 110–122, 2017.

[105] C. Szegedy, W. Liu, Y. Jia, et al., "Going deeper with convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[106] V. Tankovich, C. Hane, S. R. Fanello, Y. Zhang, S. Izadi, and S. Bouaziz, "Hitnet: Hierarchical iterative tile refinement network for real-time stereo matching," CoRR, vol. abs/2007.12140, 2020. arXiv: 2007.12140. [Online]. Available: https://arxiv.org/abs/2007.12140.

[107] C. Tomasi and S. Birchfield, "Depth discontinuities by pixel-to-pixel stereo," in Sixth International Conference on Computer Vision,(ICCV'98)(Bombay, India, 1998), 1998, pp. 1073–1080.

[108] R. Toth, Modeling and identification of linear parameter-varying systems. Springer, 2010, vol. 403.

[109] A. K. Ushani, R. W. Wolcott, J. M. Walls, and R. M. Eustice, "A learning approach for real-time temporal scene flow estimation from lidar data," in 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 5666–5673.

[110] O. Veksler, "Stereo correspondence by dynamic programming on a tree," in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), IEEE, vol. 2, 2005, pp. 384–390.

[111] R. Visvanathan, S. M. M. Syed Zakaria, K. Kamarudin, et al., "Mobile robot localization system using multiple ceiling mounted cameras," Nov. 2015, pp. 1–4. doi: 10.1109/ICSENS.2015.7370454.

[112] U. Von Luxburg, "A tutorial on spectral clustering," Statistics and computing, vol. 17, no. 4, pp. 395–416, 2007.

[113] D. Wang and K. B. Lim, "Obtaining depth map from segment-based stereo matching using graph cuts," Journal of Visual Communication and Image Representation, vol. 22, no. 4, pp. 325–331, 2011.

[114] X. Wang, H. Wang, and Y. Su, "Accurate belief propagation with parametric and non-parametric measure for stereo matching," Optik, vol. 126, no. 5, pp. 545–550, 2015.

[115] Y. Wang, J. Ji, Q. Wang, C. Xu, and F. Gao, "Autonomous flights in dynamic environments with onboard vision," in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, pp. 1966–1973.

[116] S. A. Winder and M. Brown, "Learning local image descriptors," in 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2007, pp. 1–8.

[117] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1492–1500.

[118] Z. Xu, X. Zhan, B. Chen, Y. Xiu, C. Yang, and K. Shimada, "A real-time dynamic obstacle tracking and mapping system for uav navigation and collision avoidance with an rgb-d camera," arXiv preprint arXiv:2209.08258, 2022.

[119] P. Yang, R. A. Freeman, and K. M. Lynch, "Multi-agent coordination by decentralized estimation and control," IEEE Transactions on Automatic Control, vol. 53, no. 11, pp. 2480–2496, 2008.

[120] D. Yoon, T. Tang, and T. Barfoot, "Mapless online detection of dynamic objects in 3d lidar," in 2019 16th Conference on Computer and Robot Vision (CRV), IEEE, 2019, pp. 113–120.

[121] C. Yu, Z. Liu, X.-J. Liu, et al., "Ds-slam: A semantic visual slam towards dynamic environments," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 1168–1174.

[122] Y. Yu, C. Pradalier, and G. Zong, "Appearance-based monocular visual odometry for ground vehicles," pp. 862–867, Jul. 2011. doi: 10.1109/AIM.2011.6027050.

[123] J. Zbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1592–1599.

[124] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4490–4499. doi: 10.1109/CVPR.2018.00472.

[125] Y. Zhou and C. Hou, “Stereo matching based on guided filter and segmentation,” Optik, vol. 126, no. 9-10, pp. 1052–1056, 2015.

[126] A. Z. Zhu, D. Thakur, T. ozaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, “The multivehicle stereo event camera dataset: An event camera dataset for 3d perception,” IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 2032–2039, 2018.

[127] S. Zickler, T. Laue, O. Birbach, M. Wongphati, and M. Veloso, “Ssl-vision: The shared vision system for the robocup small size league,” in RoboCup 2009: Robot Soccer World Cup XIII, J. Baltes, M. G. Lagoudakis, T. Naruse, and S. S. Ghidary, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 425–436, isbn: 978-3-642-11876-0.

Appendix A

The undistortion of the image coordinates for the fisheye lens used in the visual tracking is discussed in this Appendix. The radial distance d_r for perspective pinhole projection between the image coordinates of the incoming ray of the world point P and the principal point is defined by $d_r = \sqrt{u^2 + v^2}$, where u, v are the coordinates of the projection point in pixels (i.e., pinhole projection coordinates of P). The angle between the ray and the principal axis is denoted by $\theta = \tan^{-1}(d_r)$. The radial fisheye distortion factor Ψ_d is modeled as in [48] :

$$\theta_d = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8), \qquad (A1)$$

where $k_1, \ldots k_4$ are lens distortion parameters. The distorted image coordinates of the projected point are realized by $\bar{u} = \frac{\theta_d}{d_r} u$, $\bar{v} = \frac{\theta_d}{d_r} v$, which will be used to obtain the undistorted coordinates of the projection point (in pixels) as in

$$u = f_x(\bar{u} + \alpha\bar{v}) + c_x, \; v = f_y\bar{v} + c_y, \qquad (A2)$$

where f_x and f_y are the lens' focal lengths and c_x, c_y are the principal point coordinates (at the image center).