Freie Universität Berlin

Master's Thesis

Extension of Direct Sparse Odometry with Stereo Camera and Inertial Sensor

Author:
Sergej Mann

Advisors:
Prof. Dr. Daniel Göhring
Prof. Dr. Raúl Rojas

Mobile Robotics and Autonomous Vehicles
Dahlem Center for Machine Learning and Robotics

November 10, 2017

Declaration of Authorship

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Unterschrift: ______________________________

Datum: ______________________________

FREIE UNIVERSITÄT BERLIN

Department of Mathematics and Computer Science

Dahlem Center for Machine Learning and Robotics

Master of Science

Extension of Direct Sparse Odometry with Stereo Camera and Inertial Sensor

by Sergej Mann

Abstract. Nowadays, real-time capable visual odometry and visual simultaneous localization and mapping have become popular research topics. Since robots depend on the precise determination of their own motion, visual methods can be used for trajectory generation, localization or path planning. Different kinds of sensors can be used to tackle this — in general hard to solve — task, but it is always a trade-off between configuration effort and monetary cost of the system as well as other quality factors. Hence, it becomes increasingly popular to use cameras as sensors for the ego-motion determination of a robot.

This thesis deals with the extension of a monocular direct sparse visual odometry to a stereo direct sparse visual-inertial odometry and the evaluation of the outcome.

The depth information from a stereo camera is used to eliminate the initialization step and to pre-initialize the depth of selected pixels of keyframes. Furthermore, depth information and inertial measurements significantly robustify pose pre-initialization for new frames. Due to the known depth, the unknown scale issue is solved and the scale drift is eliminated. The experiments carried out in this work show that the extensions significantly increase both robustness and tracking accuracy.

Acknowledgements

I would like to thank all those who contributed to the development of this work.

My advisors, Prof. Dr. Daniel Göhring and Prof. Dr. Raúl Rojas, and the whole research group of Dahlem Center for Machine Learning and Robotics, who have aroused my curiosity for robotics and especially for computer vision.

My colleagues and mentors at Autonomos GmbH, especially Bennet Fischer, Daniel Seifert, Sören Hess, Philipp Holzschneider, Michael Schnürmacher, Simon Wichmann, Stefan Kaiser, Saad Ouassil Allak, Tinosch Ganjineh, Patrick Vogel and foremost Jochen Wilhelmy and Dimitri Schachmann, who immensely contributed to the creation of this thesis.

My parents for their continuous support, and above all my daughter OLIVIA and my love JANA — as well as Morpheus and Neo — for their endless support and bringing joy into my life.

Thank you very much!

Contents

List of Figures

List of Tables

List of Abbreviations

CCN	**C**onvolutional **N**eural **N**etworks
DOF	**D**egrees **o**f **F**reedom
DSO	**D**irect **S**parse **O**dometry
FoV	**F**ield **o**f **V**iew
FPGA	**F**ield-**P**rogrammable **G**ate **A**rray
FPS	**F**rames **P**er **S**econd
GPS	**G**lobal **P**ositioning **S**ystem
Hz	**H**ert**z**
IMU	**I**nertial **M**easurement **U**nit
INS	**I**nertial **N**avigation **S**ystem
MPU	**M**otion **P**rocessing **U**nit
NASA	**N**ational **A**eronautics and **S**pace **A**dministration
NCC	**N**ormalized **C**ross **C**orrelation
RANSAC	**RA**ndom **SA**mple **C**onsensus
RMSE	**R**oot **M**ean **S**quare **E**rror
SDSVIO	**S**tereo **D**irect **S**parse **V**isual-**I**nertial **O**dometry
SLAM	**S**imultaneous **L**ocalization **a**nd **M**apping
VO	**V**isual **O**dometry
VIO	**V**isual-**I**nertial **O**dometry
V-SLAM	**V**isual **S**imultaneous **L**ocalization **a**nd **M**apping

Chapter 1

Introduction

Modern vehicles have various sensors to monitor their immediate environment, e.g., Lidar, Radar and Cameras. These sensors are particularly important for autonomous driving in order to detect moving objects and avoid collisions. The determination of own motion (a.k.a. ego-motion) of a mobile robot is particularly crucial, since the robot has an influence on it by its actions [1].

Multiple different sensors — also a combination of sensors — can be used for motion estimation. An odometer and an inertial measurement unit (IMU) are often used. The odometer counts the wheel turns over time [1], and the units of an IMU are typically gyroscopes and accelerometers, which measure rotation and velocity [2]. A Lidar or global positioning system (GPS) can also be used by matching the laser scans [1] and by distinguishing their data over time.

Which sensors are used depends on the application, the cost and the desired accuracy. Each sensor has its own drawbacks. The wheel odometry is affected by wheel slip in uneven terrain, or other adverse conditions [1]. GPS and IMU suffer from drift [2], in particular the inexpensive ones, and today's laser scanners are not affordable for everyone. Especially, the inaccuracy of inexpensive sensors and the high cost of laser scanners require affordable alternative solutions. A camera is an inexpensive and lightweight sensor and offers a possible alternative.

Ego-motion estimation of a vehicle from visual input started in the early 1980s. Most of the first visual odometry (VO) researches have been motivated by the NASA for the Mars exploration program. The challenge was to provide an ability to measure the six degrees of freedom (DOF) motion for all-terrain rovers. [1]

VO describes the process of estimating the ego-motion of a camera or multiple cameras. If additionally an IMU is attached to the VO system, it is referred to as visual-inertial odometry (VIO). There are many different methods of VO and VIO. All of these methods can be classified into feature-based (indirect), appearance-based (direct) and hybrid methods [1]. Indirect methods pre-process the raw sensor measurements first to generate an intermediate representation, e.g., extracting and matching a sparse set of feature descriptors [3]. The direct methods skip the pre-processing step and directly uses the raw sensor measurements [3]. Hybrid methods combine both approaches [1].

Engel et al. [3] propose a Direct Sparse Odometry (DSO) formulation for monocular cameras, which jointly optimizes for all involved parameters, effectively performing the photometric equivalent of windowed sparse bundle adjustment.

This master's thesis has been developed in a project of Autonomos GmbH. From the point of view of this project the goal of ego-motion estimation is to provide relative, accurate and consistent camera poses as well as three-dimensional (3D) coordinates of world points (landmarks).

Figure 1.1 shows a stereo camera built and used by Autonomos GmbH. Such a stereo camera has a built-in IMU, field-programmable gate array (FPGA), and provides an Ethernet interface. The stereo camera sends over the Ethernet, among other things, the left and right image, the disparity map and inertial measurements.

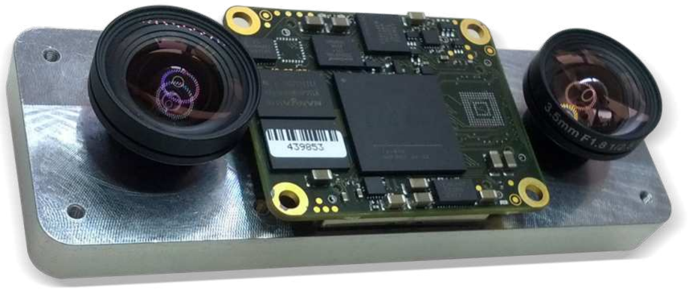


FIGURE 1.1: An example of a stereo camera used by the Autonomos GmbH.
Source: `https://www.autonomos-systems.de`

This thesis aims to provide a VIO through the extension of DSO. Furthermore, by adding depth information and inertial measurements to (1) eliminate the initialization step, (2) pre-initialize the inverse depth of keypoints, (3) provide an orientation pre-initialization for new frames, (4) increase the tracking (direct image alignment) accuracy, (5) resolve the unknown scale issue, and (6) eliminate the scale drift.

The structure of the remaining work is as follows. Chapter 2 presents a summary of previous work on visual methods for pose estimation. Chapter 3 paves the way to the theory behind stereo direct sparse visual-inertial odometry (SDSVIO). Chapter 4 describes the implementation of the extension. Chapter 5 presents the experimental setup and shows the results. Chapter 6 discusses the weaknesses and strengths of the proposed method. Chapter 7 presents the conclusions of this thesis and gives an outlook on further work.

Chapter 2

Related Work

This chapter gives an exemplary overview of some other works, which have contributed to visual odometry.

Hans Peter Moravec, a pioneer of VO, accepted the NASA challenge and introduced a motion estimation pipeline in his Ph.D. thesis [4]. His work is based on, what he termed, a “slider stereo” camera. Slider stereo is a single camera sliding on a rail. The system is reliable for short trips but is not real-time capable. The robot digitized and analyzed images after each meter. While moving horizontally, the camera grabbed nine pictures at equidistant intervals. In an image, corners are detected and matched along the epipolar lines on the other eight images, using normalized cross correlation (NCC). Outliers are removed by applying a depth consistency check. A coarse-to-fine strategy is used to determine matches again by correlation at the next robot location. The rigid body transformation is computed by aligning the triangulated 3D points, seen at two consecutive positions.

Based on Moravec’s work, Matthies and Shafer [5] took advantage of integrating the stereo geometry triangulation uncertainty as an error covariance matrix into the motion estimation step.

Several decades of research produced many non-real-time implementations [1]. The first real-time long-run capable visual odometry systems were proposed in 2004 by Nister et al. [6]. Their work has coined the term visual odometry. The term VO is based on the wheel odometry, because of the similar basic functionality. Contrary to previous works, they have detected and matched features instead of tracking them. They also compute the relative motion as a 3D-to-2D camera pose estimation problem instead of 3D-to-3D alignment problem.

Kerl et al. [7] and Engelet et al. [8, 9, 3] push direct methods forward. Kerl et al. [7] present a direct dense visual simultaneous localization and mapping (V-SLAM) method for RGB-D cameras that minimizes both the depth and the photometric error over all pixels. They also propose a keyframe selection based on entropy similarity measurements. Engel et al. [8] propose a direct semi-dense VO for monocular cameras, with the main idea to continuously estimate a semi-dense inverse depth map for the current image, which in turn is used to track the motion using direct dense image alignment.

In [9], Engel et al. demonstrate the large-scale direct monocular V-SLAM (LSD-SLAM), another direct method for monocular cameras. LSD-SLAM is reliable for large-scale trips and creates a consistent map of the environment. The proposed method involves a probabilistic depth noise model and aims to recognize scale drift. In their further work, Engel et al. [10] extended LSD-SLAM to stereo LSD-SLAM (S-LSD-SLAM). They took advantage of depths from both stereo-view (capture images from different cameras at same time) and multi-views (capture images from same camera at different times), and propose an approach to handle aggressive brightness changes between images. Usenko et al. [11] extend S-LSD-SLAM by tightly incorporating an IMU that simultaneously estimates camera pose, velocity, and IMU biases, minimizing a combined photometric and inertial energy function. Zhu [12] contributes three improvements to S-LSD-SLAM: (1) a dual Jacobian optimization scheme to avoid local optima and improve the accuracy, (2) the gradient-based keypoint representation to be robust to changes in illumination, (3) a joint direct VO energy function to incorporate the information from multiple images.

Engel et al. [13, 3] present a simple photometric calibration method and direct sparse VO. They aim in [13] to improve the input in a preprocessed manner for direct approaches by providing photometric calibration, which contains the camera response function and pixel-wise attenuation factors. In [3], they propose a direct sparse VO formulation for monocular cameras. In contrast to existing direct methods, [3] jointly optimizes for all involved parameters, effectively performing the photometric equivalent of windowed sparse bundle adjustment. The proposed model integrates a full photometric calibration, accounting for lens vignetting, non-linear response functions and exposure time. Furthermore, [3] shows that the proposed direct formulation outperforms the state-of-the-art feature-based (indirect) monocular SLAM method ORB-SLAM [14].

During the emergence of this master's thesis a similar work “Stereo DSO” from Wang et al. [15] has been published. In contrast to this thesis, [15] is independent of other sensors and uses the stereo image pair as input to verify the selected points and assist the depth initialization.

Methods based on machine learning [16, 17], especially convolutional neural networks (CNN), are becoming increasingly common nowadays and could one day become the new state-of-the-art.

Chapter 3

Fundamentals

This chapter paves the way to the theory behind Stereo Direct Sparse Visual-Inertial Odometry (SDSVIO).

3.1 Visual Odometry

Vehicles have a device called odometer on the dashboard, which indicates the length of the travelled distance. In robotics, odometry refers to estimating the entire trajectory of a mobile robot, not just the traveled distance. When one or more cameras are used for trajectory determination, it is called VO. The trajectory determination contains the motion estimation (also referred to as ego-motion estimation) step, which estimates the pose change between the current and the previous camera. Six parameters have to be determined in 3D space, since a pose contains six DOF. A camera pose consists of a position $= (x, y, z)^\intercal$ and an orientation $= (\alpha, \beta, \gamma)^\intercal$ with respect to the previous frame and can also be written as stacked vector: $\xi = (x, y, z, \alpha, \beta, \gamma)^\intercal$. There is more than one way to represent the six pose parameters. The position is described here along the three axes in the Cartesian coordinate system. Three Euler angles α, β and γ describe the orientation, also known as roll, pitch and yaw. In general, to obtain the final orientation, a rotation around the x-axis is applied by α degree, then around the y-axis by β degree and finally around the z-axis by γ degree.

VO describes the process of estimating the ego-motion of a camera or multiple cameras, where the pose of the camera is incrementally determined. VO aims to achieve local consistency of the trajectory. An optimization over the last n poses (referred to as windowed bundle adjustment) is often used to obtain a more accurate trajectory [1]. In general, sequentially ordered images of geometrically calibrated cameras are used. Direct approaches also benefit from photometric calibration [3, 13, 18]. When only one camera is used, it is referred to as monocular VO. When using several cameras, it is called stereo VO. If additionally an inertial sensor is attached to the VO system, it is referred to as visual-inertial odometry (VIO).

Both monocular and stereo approaches have their benefits and drawbacks. In contrast to stereo approaches, where the absolute scale can be determine by 3D measurements, in the monocular case the motion can only be determined up to an unknown scale factor. The monocular methods require an initialization step, often setting the distance between the first two poses to one. Subsequently, the relative scale and pose of the new images are determined with respect to the first two images. In case of small motions, the monocular scheme shows more drift than the stereo scheme [1]. For the reasons mentioned above, the stereo methods are often more robust. However, if the distance to the scene is much larger than the baseline, stereo VO degenerates into the monocular case and monocular methods must be used [1]. The same case occurs when the distance to the scene is too small. Depending on the application, the weight and size of the VO system also plays a decisive role. A very small drone robot (e.g. a robobee) is not always able to carry a stereo system, and again monocular methods must be used.

In fact, the unknown scale factor is not a problem in practice if everything else within the frame of reference is correct. But that is not the case, because the relative scale of a frame is only known in relation to its predecessor. Any inaccuracy in the measurement is propagated and this may cause the scale to successively change over time. This core problem of monocular VO is called scale drift. The trajectory is more and more different from the ground truth. Techniques such as windowed bundle adjustment or a combination with other sensors, such as inertial sensor, can reduce the scale drift [1].

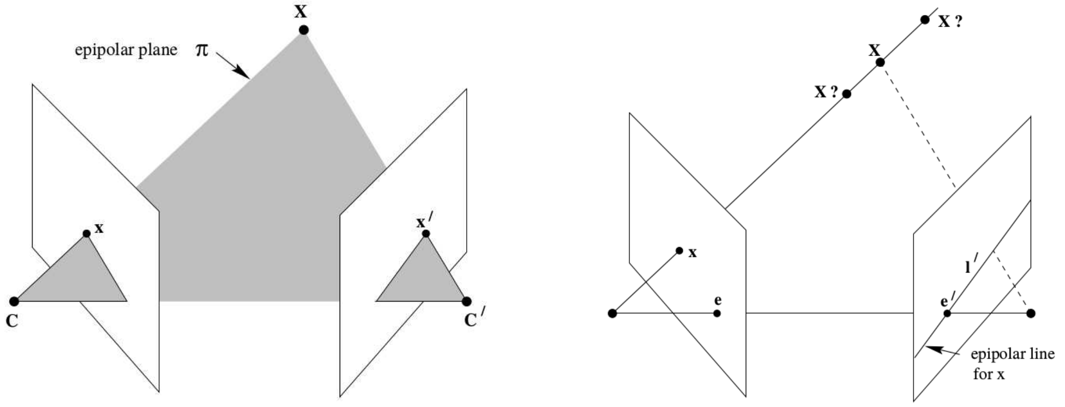
For the VO to work effectively, a number of conditions must be applied. Sufficient illumination is required as well as a static scene with sufficient texture, and sufficient scene overlap of consecutive frames. [1]

3.2 Stereo Vision

It is interesting to see the difference between using a stereo camera which is simply two separate cameras in space, and using only a single camera. When the 3D world is projected onto a 2D image, one dimension — the distance — is completely lost. The lost dimension can be reconstructed using a stereo camera, similar to the human vision.

Figure 3.1a illustrates the theory of depth reconstruction based on epipolar geometry. It is obvious that the involved points, world point $\mathbf{X}$, both camera centers $\mathbf{C}$ and $\mathbf{C}'$, span a plane π and that the projected points $\mathbf{x}$ and $\mathbf{x}'$ of $\mathbf{X}$ are in this plane. Furthermore, as illustrated in figure 3.1b, the corresponding pixels in both cameras are needed for robust depth reconstruction.

Often an axis-parallel stereo system is used, which is characterized by a parallel orientation of the optical axes of both cameras, i.e., the cameras are only moved horizontally and their coordinate systems are not rotated against each other. The two optical centers are only horizontally shifted. In practice, the cameras can only be approximately aligned in parallel. It therefore requires a virtual alignment of the cameras in order to convert them into an axis-parallel camera system. Rectification accomplishes that. The aim of the rectification is that the so-called epipolar lines (see figure 3.1b) are all horizontal and thus the corresponding pixels are on the same image line. By using rectification, simpler processing structures



(A) Epipolar plane (B) Epipolar line

Figure 3.1: Illustration of epipolar geometry [19]

and thus more efficient pixel correspondence analysis can be achieved. Since the image rows are located on the same image line after the rectification, the different perspectives of the cameras, with respect to the world point X, lead to a pure horizontal offset in the image. This difference is called disparity d and is used to compute the distance of $\mathbf{X}$ as follows:

$$z = \frac{B \times f}{d} \tag{3.1}$$

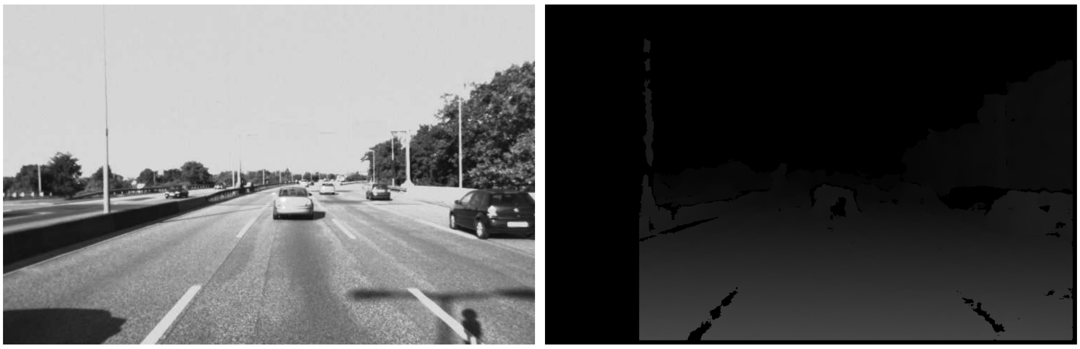
Where z is the distance from the left camera center, B describes the distance between the camera centers, called baseline, f represents the distance from camera center to sensor plane, called focal length.

A stereo camera mounted on a vehicle not only provide images, but also distance information available in the form of disparity maps. A stereo camera system consists of two to each other shifted cameras, which observe a scene. With such a system, two perspectively different images of a scene are captured simultaneously. Due to the fixed and known geometry between the cameras, a depth reconstruction is possible.

In the context of this thesis, the experimental testbed has been equipped with a stereo camera used to provide images and distance information in form of disparities. The stereo correspondence analysis is performed on the built-in FPGA and assumed as given for this work. Figure 3.2 shows a rectified camera image with the corresponding disparity image.

3.3 Inertial Measurement Unit

An IMU typically uses accelerometers to measure velocity and gyroscopes to measure rotation — and there are usually several of each inside [2]. Also MPU-6050 [20] — built-in the used stereo camera — is using a gyroscope and a accelerometer.



(A) rectified image (B) disparity image

FIGURE 3.2: The left image of a stereo camera and the corresponding disparity image.
The black pixels in the disparity image indicates where no sufficient information is available for pixel correspondence analysis. The grayscale encode disparity values: bright = near and dark = far.

The MPU-6050 also includes a motion processing unit (MPU), which supports 3D motion processing and gesture recognition algorithms. The MPU collects 3-axis accelerometer and 3-axis gyroscope measurements while synchronizing data sampling at a user defined rate. [20, p. 11]

In addition to accelerometer and gyroscope measurements, the MPU calculates the orientation in the space of the IMU. The accelerometer measurements are used to detect orientation in relation to the gravity. The gyroscope is used to detect rotation changes in space. However, it is not documented by the manufacturer how this is calculated in detail and is therefore assumed to be as given for this work.

3.4 Direct Sparse Odometry

This section gives an overview over the functionality of DSO [3]. DSO is a novel sparse and direct formulation of monocular VO developed at the Technical University of Munich.

DSO operates directly on image intensities and is able to sample a sparse pixel set from all image regions that provide an intensity gradient, i.e., it does not depend on hand-crafted feature descriptors, so it does not need to perform descriptor extraction or descriptor matching. It is based on minimizing the photometric error of the sparse pixel set between frames, instead of minimizing the reprojection (geometric) error, e.g., like bundle adjustment. The camera tracking is done by direct image alignment using the coarse-to-fine strategy. DSO optimizes for all involved parameters, camera poses, camera intrinsics, and geometry — represented as inverse depths. Essentially, performing the photometric equivalent of windowed sparse bundle adjustment.

The tracking uses the coarse-to-fine strategy and the keypoint selection operates on image gradients. Direct approaches took advantage of photometric undistortion. As an early preprocessing step, each image is photometrically undistorted and image pyramids, as well as gradient pyramids are produced.

3.4.1 Formulation

For simplicity, the method has been formulated for a pinhole camera model. It is assumed that the images are geometrically undistorted in a preprocessing step.

The projection is denoted by $\Pi_{\mathbf{c}} : \mathbb{R}^3 \to \Omega$ and reprojection by $\Pi_{\mathbf{c}}^{-1} : \Omega \times \mathbb{R} \to \mathbb{R}^3$, where $\mathbf{c}$ denotes the camera intrinsics. Camera poses are represented as transformation matrices $\mathbf{T}_i$, transforming a point from the world frame into the camera frame.

Photometric Calibration

The used image formation model comes from [13], it takes into account a non-linear response function $G : \mathbb{R} \to [0, 255]$ and lens attenuation (vignette) $V : \Omega \to [0, 1]$. The combined model is given by

$$I_i\left(\mathbf{x}\right) = G\left(t_i V(\mathbf{x}) B_i(\mathbf{x})\right), \tag{3.2}$$

where t_i is the exposure time, B_i is the irradiance and I_i the pixel intensity in frame i. The photometric correction model is given by

$$I_i'\left(\mathbf{x}\right) = t_i B_i(\mathbf{x}) = G^{-1}(I_i(\mathbf{x})) \times V^{-1}(\mathbf{x}). \tag{3.3}$$

Where I_i' is the photometrically undistorted image. However, in the further course of the work, I_i refers to the photometrically undistorted image I_i'.

The response function of an image sensor represents an association of received irradiance values by a photocell during the exposure time to the respective pixel value. To undo this, the inverse response function is used so that the pixel value is linear to the received irradiance value. The vignette describes the lens effect, which causes photocells at the corners of the image sensor to receive less irradiance than central photocells. The vignette represents a pixel-wise attenuation factor. Since a division is more computationally expensive than a multiplication, the inverse vignette is used instead.

Photometric Error

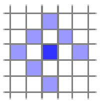


FIGURE 3.3: The residual pattern $\mathcal{N}_{\mathbf{p}}$

The photometric error over all active keyframes and all active points is given by

$$E_{photo} = \sum_{i \in \mathcal{F}} \sum_{\mathbf{p} \in \mathcal{P}_i} \sum_{j \in \text{obs}(\mathbf{p})} E_{\mathbf{p}j}. \tag{3.4}$$

Where i runs over all keyframes $\mathcal{F}$, $\mathbf{p}$ over all points $\mathcal{P}_i$ in frame i, j over all frames obs($\mathbf{p}$) where the point $\mathbf{p}$ is visible. $E_{\mathbf{p}j}$ denotes the photometric error with residual pattern $\mathcal{N}_\mathbf{p}$ of $\mathbf{p}$, as shown in figure 3.3, in reference frame I_i and observed in another frame I_j is formulated as

$$E_{\mathbf{p}j} = \sum_{\hat{\mathbf{p}} \in \mathcal{N}_\mathbf{p}} w_{\hat{\mathbf{p}}} \left\| I_j(\hat{\mathbf{p}}') - b_j - \frac{t_j e^{a_j}}{t_i e^{a_i}} \left(I_i(\hat{\mathbf{p}}) - b_i \right) \right\|_\gamma . \tag{3.5}$$

Where $\| \, \|_\gamma$ denotes the Huber norm, t_i, t_j the exposure times of the images I_i, I_j, e^{a_i}, e^{a_j} the multiplicative part and b_i, b_j the additive part of a brightness transfer function of the images I_i, I_j and $\hat{\mathbf{p}}'$ represents the projected point position of $\hat{\mathbf{p}}$ in I_j with inverse depth $d_\mathbf{p}$, given by

$$\hat{\mathbf{p}}' = \Pi_\mathbf{c} \left(\mathbf{R} \Pi_\mathbf{c}^{-1} \left(\hat{\mathbf{p}}, d_\mathbf{p} \right) + \mathbf{t} \right) \tag{3.6}$$

with

$$\begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix} = \mathbf{T}_j \mathbf{T}_i^{-1}, \tag{3.7}$$

and $w_{\hat{\mathbf{p}}}$ is a gradient-dependent weighting, that down-weights pixels with high gradient, given by

$$w_{\hat{\mathbf{p}}} = \frac{c^2}{c^2 + \| \nabla I_i \left(\mathbf{p} \right) \|_2^2} \tag{3.8}$$

with some constant c.

For clarify, each point contributes with $\|\mathcal{N}_\mathbf{p}\| = 8$ residuals to the error function to improve the robustness against motion blur.

In summarize, the error depends on: (1) the inverse depth $d_\mathbf{p}$ of each point, (2) the camera intrinsics $\mathbf{c}$, (3) the pose of the involved frames $\mathbf{T}_i$, and (4) their brightness transfer function parameters a_i, b_i, a_j, b_j.

The additional dependency of each residual on the pose of the host frame represents the only difference to the classical reprojection error, i.e., each term depends on two instead of just one frames.

If the exposure times, as well as the camera response function are known, the modeling of the affine brightness changes is omitted. Than the equation 3.5 —

which models the direct image alignment in combination with an affine brightness transfer function – only represents the direct image alignment part by

$$E_{\boldsymbol{p}j} = \sum_{\hat{\mathbf{p}} \in \mathcal{N}_{\mathbf{p}}} w_{\hat{\mathbf{p}}} \left\| I_j(\hat{\mathbf{p}}') - I_i(\hat{\mathbf{p}}) \right\|_{\gamma} . \tag{3.9}$$

3.4.2 Front-End

The front-end is the part of the algorithm that decides (1) which points and frames are used, (2) about the visibility of points in other frames, (3) about outlier removal and occlusion detection, and (4) about marginalization of points and frames.

Frame Management

DSO always keeps a fixed size window of active keyframes. Every new frame is initially tracked with respect to the newest keyframe. Then, DSO decides whether the frame is discarded or selected as a new keyframe. When a new keyframe is created, the photometric error is optimized. Afterwards, the marginalization is applied.

Initial Frame Tracking: All active points from all active keyframes are projected onto the new selected keyframe, this always creates a semi-dense depth map. The coarse tracking also pre-calculates the pose — based on previous motion and additionally on a constant motion model — of the current frame by minimizing the photometric error during the direct image alignment. This is done in a combination with coarse-to-fine method with respect to only the newest keyframe.

Keyframe Creation: Combines the following three criteria to decide if a new key frame is required. A new keyframe should be taken when:

1. the field of view changes sufficiently.
2. translation causes occlusions and disocclusions.
3. the exposure time changes significantly.

Keyframe Marginalization: Old keyframes are removed by marginalization using the Schur complement [3, 21]. The marginalization works as follows:

1. Always keep the latest two keyframes.
2. Frames with less than 5% of their points visible in the latest keyframe are marginalized.
3. If more than a given number of frames are active, except the latest two keyframes, marginalize the one which maximizes a distance score.

A keyframe is marginalized by first marginalizing all hosted points, and then the frame itself.

Point Management

DSO always keeps a fixed number of active points, equally distributed across space and active frames. First, candidate points are identified. Candidate points are traced individually in subsequent frames and not immediately added into the optimization. Candidate points are reserved for activation when new points are needed. The tracing generates a coarse depth value which will serve as initialization.

Candidate Point Selection: The image is divided into blocks of 32×32 pixels. For each block all points with a gradient magnitude greater than a fixed threshold plus the median absolute gradient are selected.

Candidate Point Tracing: Point candidates are traced in subsequent frames using a discrete search along the epipolar line. For the best match the depth and associated variance is refined. The depth and variance is used to constrain the search interval for the subsequent frame.

Candidate Point Activation: New point candidates are activated to replace the marginalized ones. All active points and candidate points are projected onto the most recent keyframe. It then activates candidate points which maximize the distance to any existing point.

Outlier and Occlusion Detection

DSO aims to identify and remove potential outliers as early as possible, since the image data generally contains much more information than can be used in real-time. Points for which the depth minimum is not sufficiently distinct during candidate tracing, are permanently discarded. Point observations for which the photometric error surpasses a — continuously adapted (with respect to the median residual in the respective frame) — threshold are removed.

Chapter 4

Implementation

This chapter describes the extension of DSO with stereo camera and inertial sensor data.

The VO application implemented in this work is at the beginning of a processing chain for map creation of short vehicle trips. The results of this work are used in applications such as ground mesh estimation and ground projection.

For the extension, disparities and inertial measurements are used as additional information provided directly by the stereo camera. The right image of the stereo camera is not used. The inertial measurements are not integrated in to the optimization.

In DSO, a point is parametrized by the inverse depth in the reference frame, so during this chapter, depth refers to inverse depth.

4.1 Stereo Camera Integration

Depth information is used to eliminate the initialization step, which is required in the monocular case to give the world a scale, and to pre-initialize the depth of candidate points.

4.1.1 Initialization

Monocular approaches can determine the motion up to an unknown scale factor. Therefore, the distance between the first two poses in the internal unit is often set to 1. DSO uses a coarse initialization step that processes multiple frames and constantly refines the pose of the subsequent frames and the depth values of the semi-dense point set by minimizing the photometric error. Afterwards, a sparse subset of the semi-dense set of point is randomly selected and provides the necessary information for the described procedure in section 3.4.2 together with the first and last frame of the initialization step.

However, using a stereo camera makes obtaining initial depth values more straightforward thus makes the initialization step unnecessary. Instead, candidate points are determined for the first incoming frame, and all points with existing depth information are activated.

4.1.2 Candidate Point Selection and Tracing

Before a candidate point is activated, its depth is continuously refined by tracing in subsequent non-keyframes. In the monocular case, the candidate points are initialized with a depth range $[0, \infty]$, that corresponds to a big variance.

The used stereo camera provides dense disparity maps, so candidate points are only selected if a disparity value for the point is available. The depth range is restricted by a small area around the given depth value.

4.1.3 Frame Tracking

When a new keyframe is selected, all active points from all active keyframes are projected onto it. That produces a semi-dense depth map. Based on this semi-dense depth map related to the newest keyframe, subsequent non-keyframes are coarse tracked.

In the extension, a combination of projected depth values from multi-views and depth values from the stereo-view are used. Primarily depth values from the stereo-view and secondarily from the multi-views. If no depth information on the projected pixel from the stereo-view is available, only the obtained depth values from multi-views are used.

4.2 Inertial Sensor Integration

The orientation of the inertial unit supports the coarse tracking. Motion detection based on the inertial measurements is applied to reduce the computational consumption and the number of keyframes if no movement is detected.

4.2.1 Constant Motion Model

The pre-initialization of new frames is based on previous motion. The assumption is that the current motion is similar to the previous one.

If the current root-mean-square error (RMSE) for a frame is more than twice the previous frame, it is assumed that the direct image alignment failed and the motion differs greatly from the previous one and attempt to recover by initializing with up to 27 different small rotations in different directions is made. To make direct image alignment more robust, the rotation of the assumed motion is overwritten with the inertial sensor orientation. In theory, only the translational part has to be determined in this way.

4.2.2 Motion Detection

Based on all measurements, angular velocity, linear acceleration, and orientation the inertial sensor is used for motion detection. Motion is assumed when one of the values exceeds its corresponding threshold value. To avoid false negatives the threshold values are sufficient small. Combining both threshold-based signals, inertial and visual, detects when there is no ego-motion. When this occurs, only the inertial measurements are monitored, e.i., no direct image alignment is performed and thus no keyframes are selected until the inertial sensor reports motion.

This simple strategy helps, among other things, when standing at the traffic lights and while waiting at crossroads.

Chapter 5

Evaluation

This chapter presents the recording setup of the own stereo dataset and the results obtained by SDSVIO (Stereo Direct Sparse Visual-Inertial Odometry) on trips recorded from a vehicle driving outdoors with challenging real-word scenarios, which is referred to as Stereo Dataset. To be comparable with other approaches an evaluation is also conducted on the popular KITTI dataset [22]. Both datasets provide synchronized and rectified stereo images.

5.1 Stereo Dataset

For the experiments, a stereo camera and two monocular cameras are mounted on a vehicle roof and are aligned with the driving direction. All cameras are triggered simultaneously to capture images. The setup shown in figure 5.1 has also an Applanix GPS/INS and a wheel odometry installed. All cameras are extrinsically calibrated to the vehicle. Therefore, the fusion of Applanix GPS/INS data and wheel odometry is considered as ground truth. The poses of the ground truth are interpolated and transformed into the stereo camera coordinate frame.

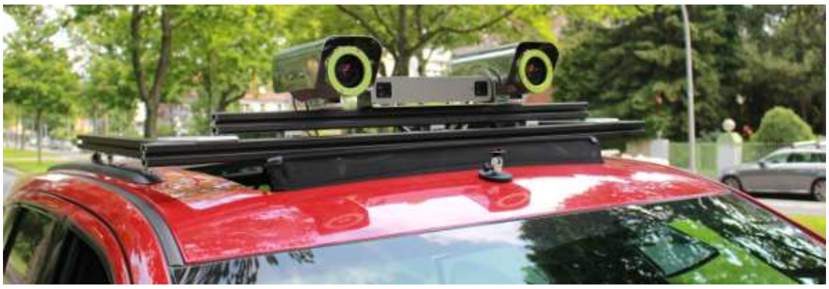


Figure 5.1: A monochrome stereo camera system (in the center) and two monocular color cameras (on the sides) are used for experiments in this work, mounted on a vehicle roof and aligned to the driving direction. Only data from the stereo camera was used for the evaluation.

All cameras run with 30 Hz and the IMU with 40 Hz. IMU data is not synchronized with the stereo frames. Approximate time synchronization is used to match the corresponding IMU data to the stereo frames, effectively performing the nearest neighbor algorithm on the time line.

The stereo camera has built-in monochrome cameras with a resolution of 752 x 480 pixels and a baseline of 25 centimeters. The dense disparity map provided by the stereo camera is used to calculate the depth information.

The two monocular color cameras with a baseline of 48 centimeter and a resolution of 1920 x 1200 pixels can also be used as a stereo camera, but they are only triggered simultaneously. That's why it is not guaranteed that they provide the images with the same timestamp. Since they, e.g., have an independent auto exposure control, and the extrinsic calibration is also not to be trusted after a few recording days, in contrast to the stereo camera. However, only data from the stereo camera was used for the evaluation.

The results are compared with DSO. For the sake of fairness, DSO is initialized with the correct scale by providing the first ten depth images but without IMU data.

The following shows four trips with path plots and absolute translation errors. The path plots show the path of ground truth, DSO and SDSVIO. The absolute translation error plots present the absolute distance of DSO and SDSVIO to the ground truth on all three axes and also shows the euclidean distance corresponding to the traveled distance based on the ground truth.

Trip *tunnel*

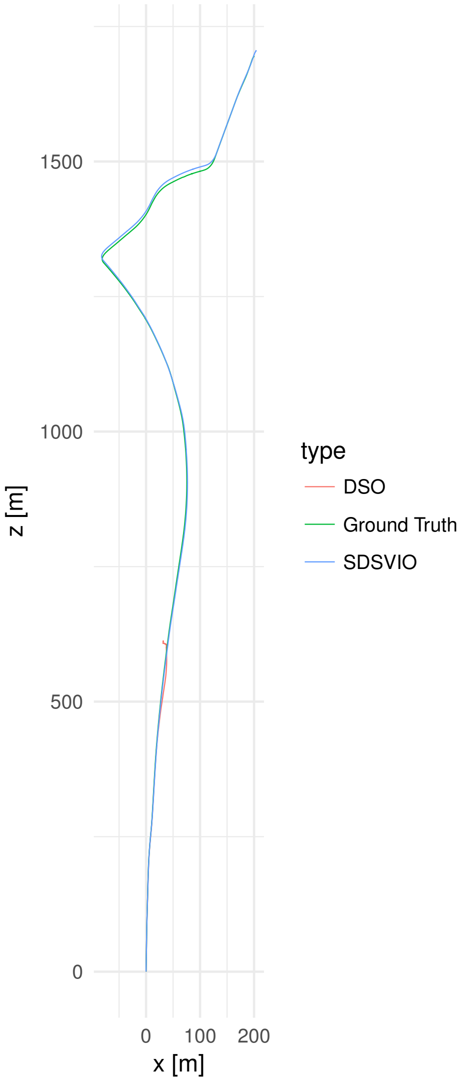


FIGURE 5.2: Path comparison of the trip *tunnel*

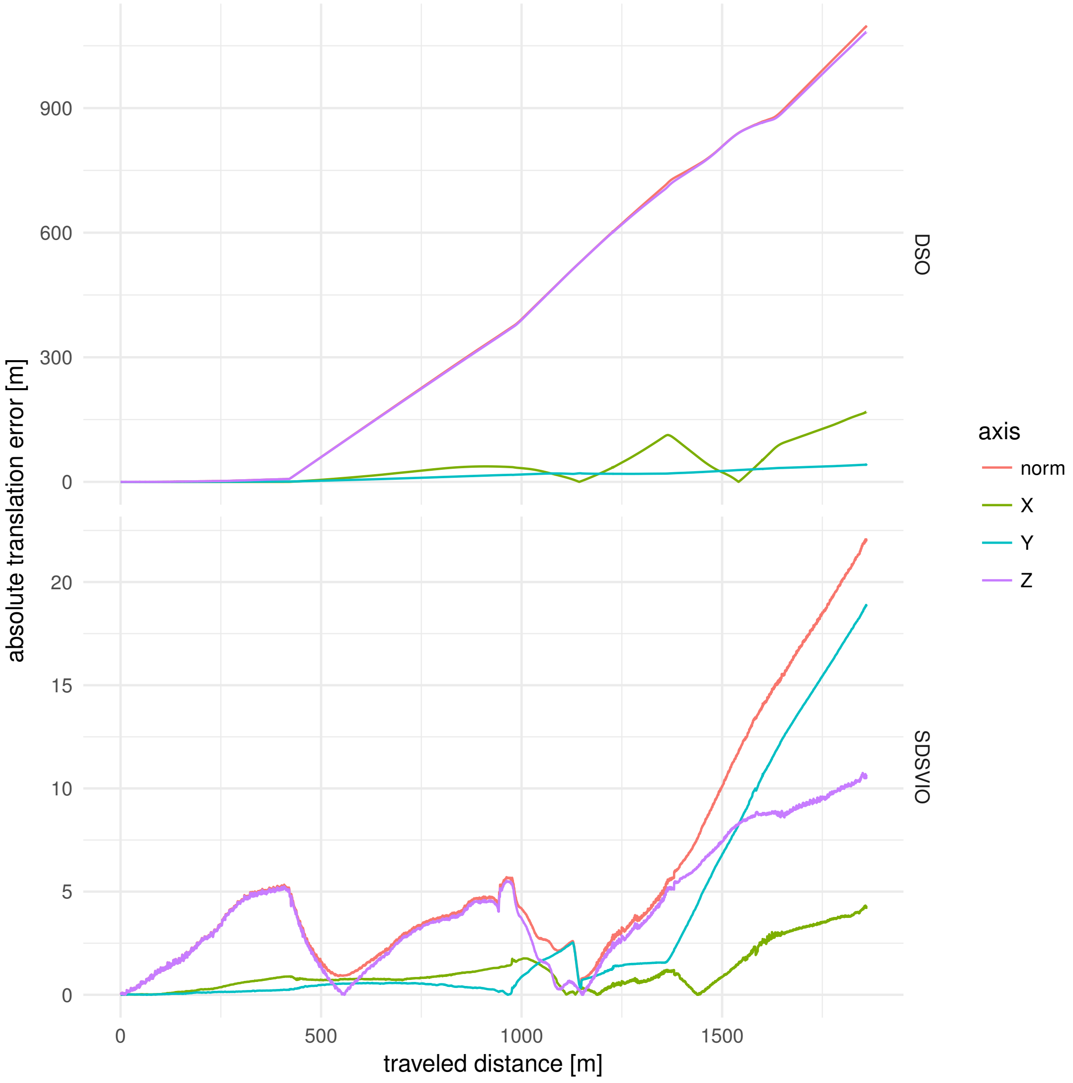


Figure 5.3: Absolute translation error of the trip *tunnel*

Trip *engler-loop*

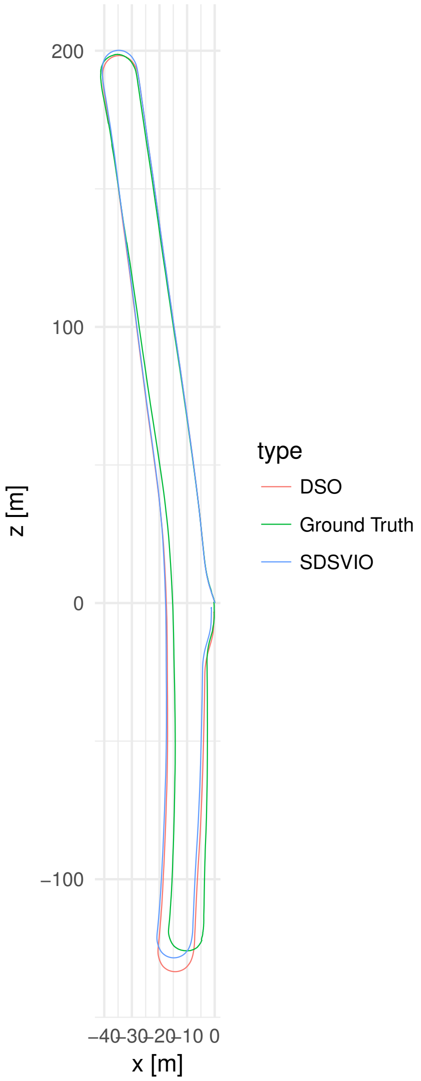


FIGURE 5.4: Path comparison of the trip *engler-loop*

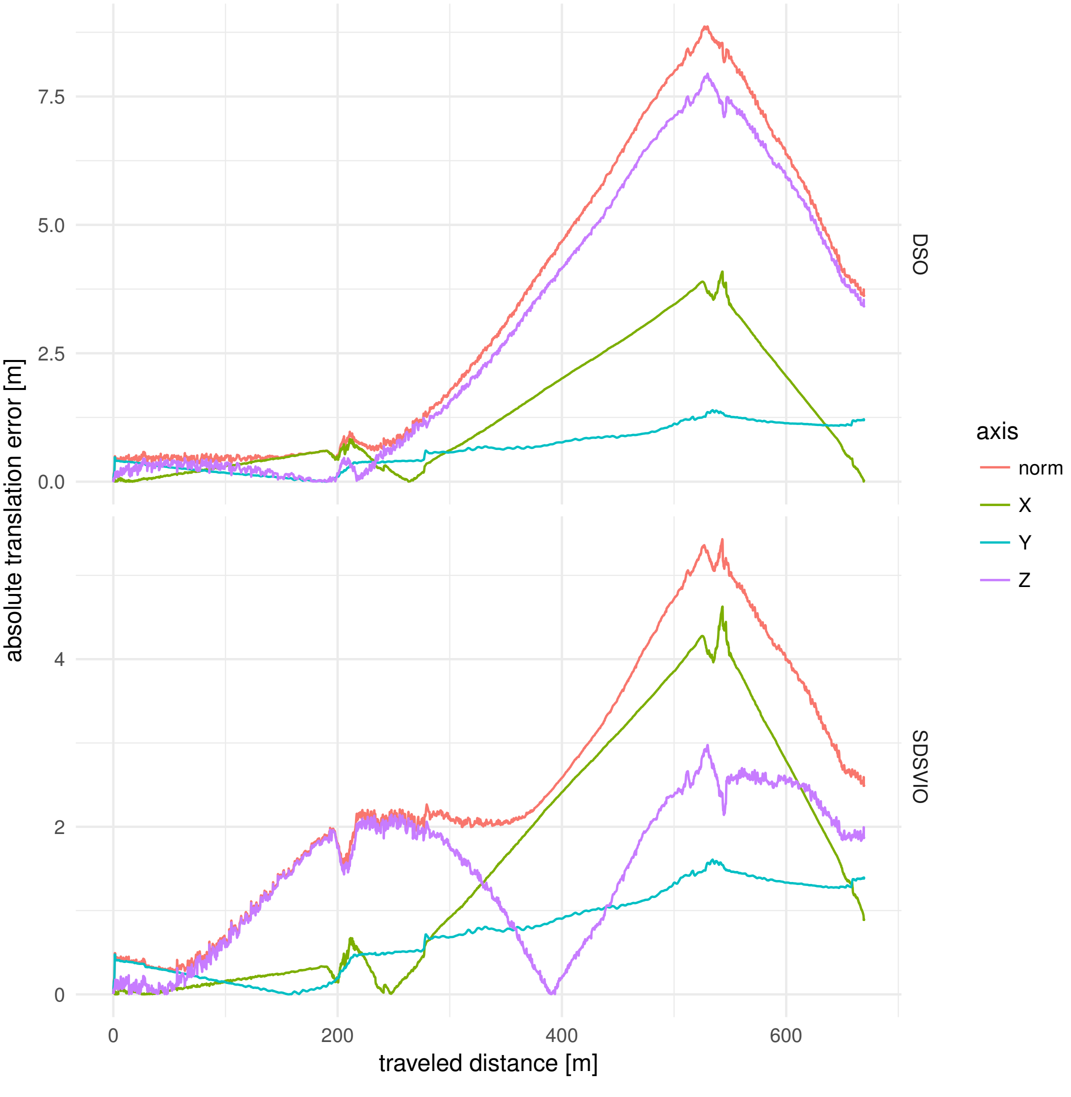


FIGURE 5.5: Absolute error of the trip *engler-loop*

Trip *engler-eightloop I*

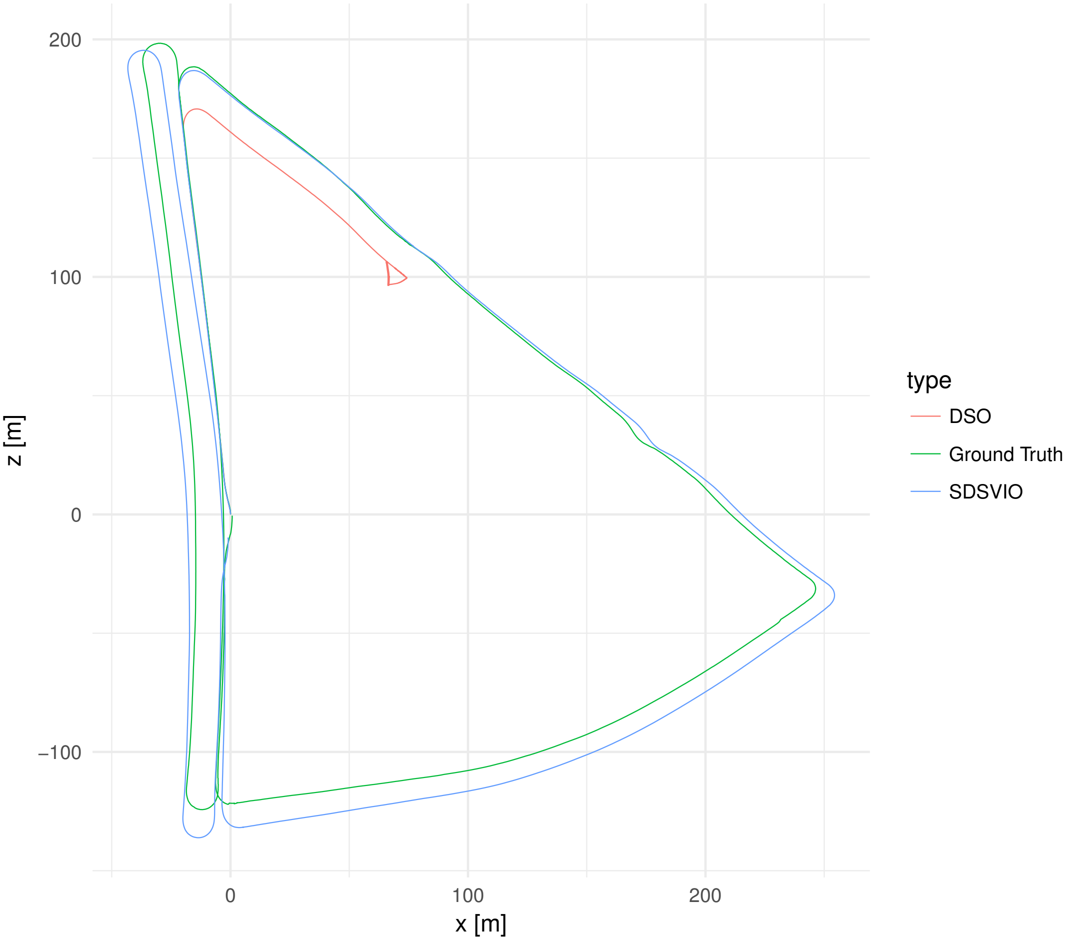


FIGURE 5.6: Path comparison of the trip *engler-eightloop I*

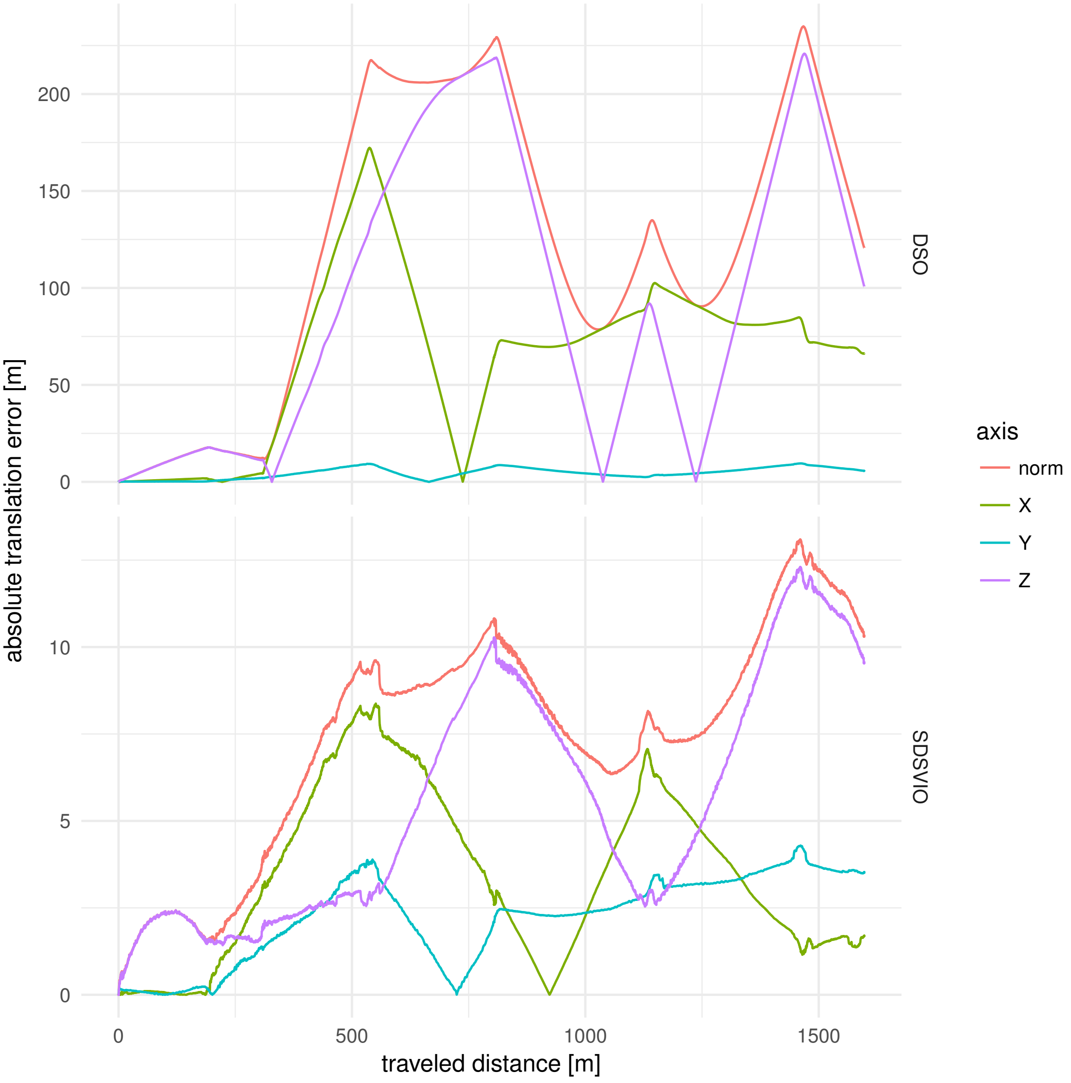


FIGURE 5.7: Absolute error of the trip *engler-eightloop I*

Trip *engler-eightloop II*

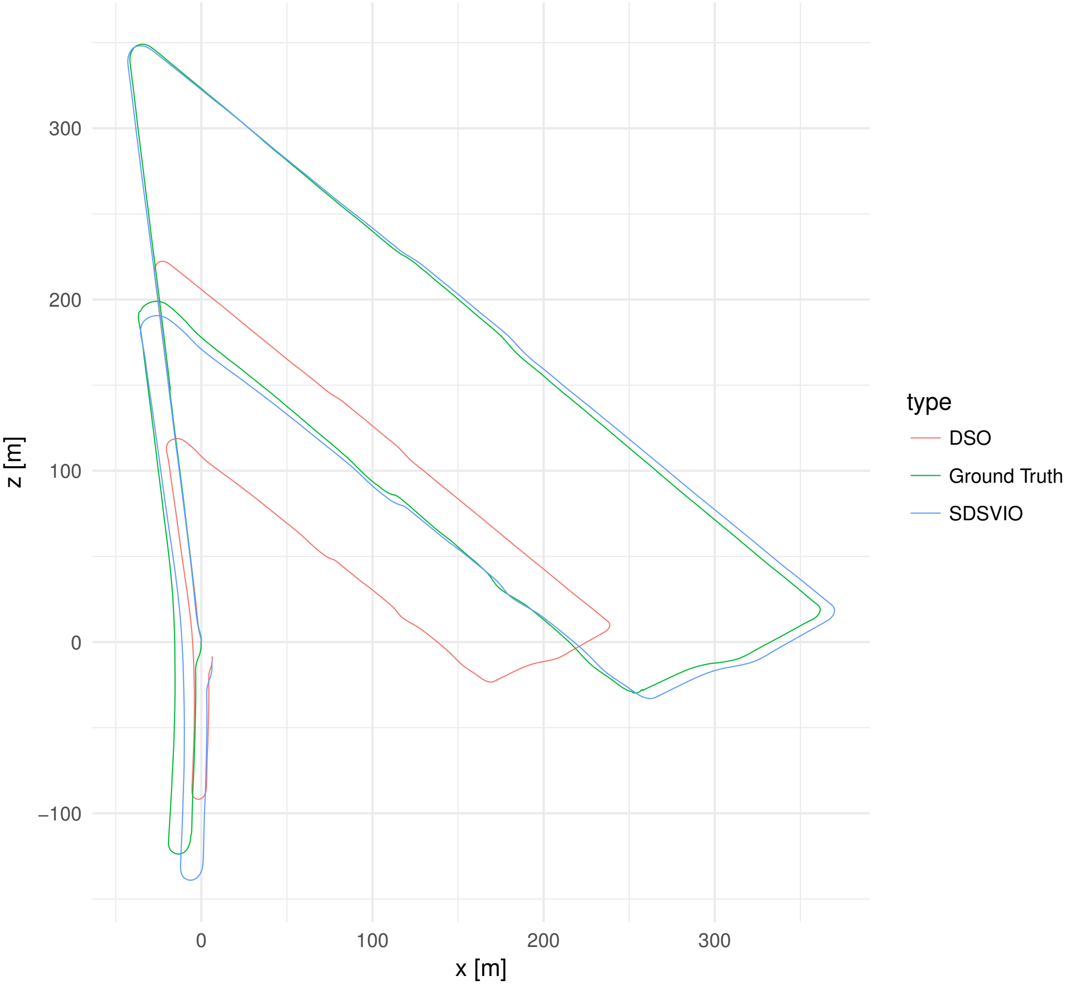


FIGURE 5.8: Path comparison of the trip *engler-eightloop II*

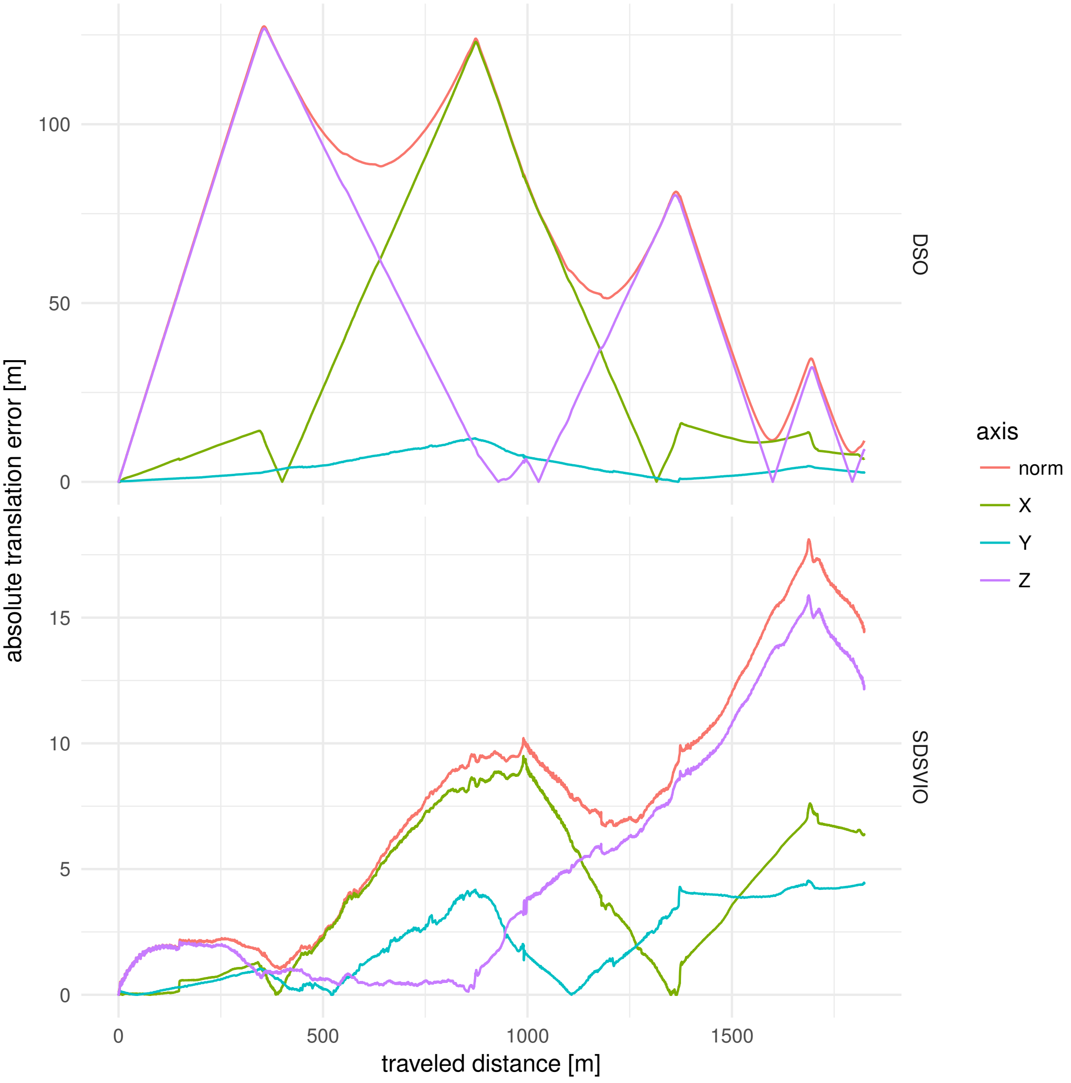


FIGURE 5.9: Absolute error of the trip *engler-eightloop II*

5.2 KITTI Visual Odometry Benchmark

The training sequences of the KITTI Visual Odometry Benchmark [22] are also used for evaluation. The grayscale images are captured with 10 Hz, a resolution of 1392 x 512 pixel and opening angle of 90° × 35° and the baseline is roughly 54 centimeters [22]. Since the method implemented in this work is based on inertial measurements, the sequences of the KITTI Raw Dataset [23] corresponding to each odometry sequence are used to obtain the inertial measurements. The inertial data is synchronized with the stereo frames and gaps are linearly interpolated [23]. In addition, inertial data has been transformed into the camera coordinate frame. However, the raw sequence *2011_09_26_drive_0067*, which corresponds to the odometry sequence 03, was not available on the KITTI website at the time of writing this work. The sequence 03 is therefore analyzed with depth information only, i.e. without inertial measurements.

For the calculation of depth information OpenCV[1], an open source computer vision library, is used. The block matching algorithm is used, because it is compatible to the results of the used stereo camera.

The KITTI dataset does not provide photometric calibration. To counteract this drawback, it has been decided to analyze the influence of using a high pass filter. This idea is inspired by [12], in particular the contribution of the gradient-based keypoint representation to be robust to changes in illumination. To be precise, the magnitude of both gradient directions is used. Note, in case of gradient images as input, a second order image gradient is applied during the pixel selection.

The KITTI evaluation computes translational and rotational errors for subsequences of length 100,. . . ,800 meters. For the sake of fairness, DSO is again initialized to the correct scale by providing the first ten depth images, but without IMU data. In Table 5.1 the average results of DSO and SDSVIO are summarized. Both are executed with grayscale images (referred as "grayscale") and gradient images (referred as "gradient"). Figure 5.10 shows the average of translational and rotational errors of SDSVIO on gradient images for different trajectory lengths and driving speeds over all test sequences. A more detailed comparison of each sequence can be found in the supplementary appendix A.

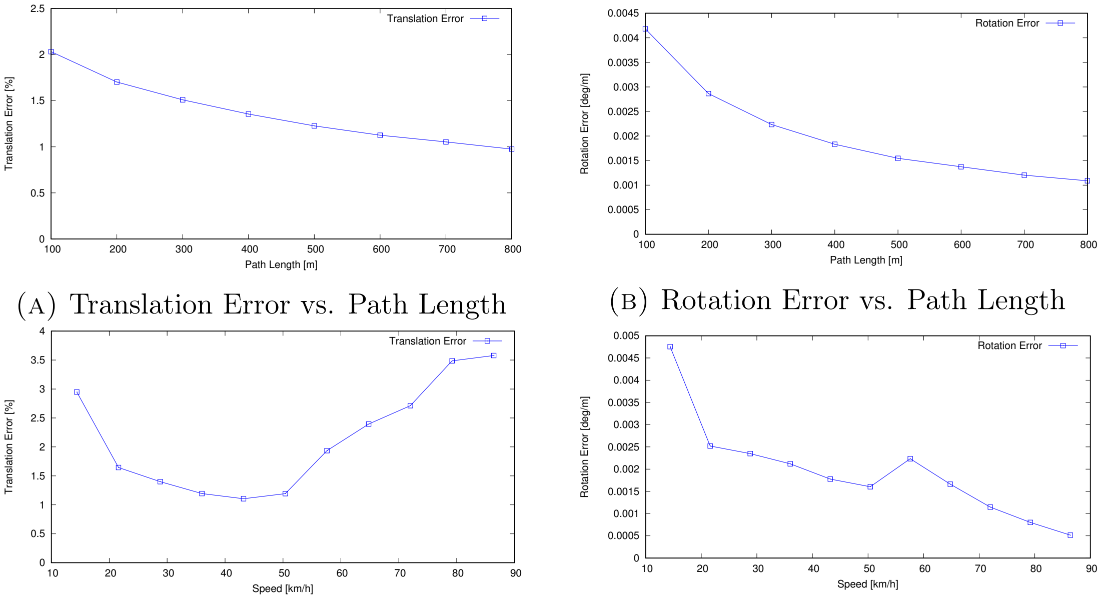
Furthermore, the comparison of the VO accuracy of different methods on the KITTI training set is presented in Table 5.2. The proposed SDSVIO on gradient images is compared to "Stereo DSO" [15], S-LSD-VO [10] and ORB-SLAM2 [24]. The first method "Stereo DSO" is similar to this master thesis, since both works are based on DSO. The other methods, S-LSD-VO and ORB-SLAM2, are currently the state-of-the-art direct and indirect stereo V-SLAM methods. The results for S-LSD-VO are cited from [10]. S-LSD-VO means S-LSD-SLAM, but only performing loop-closure in a small window of the latest frames, this turns S-LSD-SLAM into a VO [10]. Both results for "Stereo DSO" and ORB-SLAM2 are cited from [15]. The authors of [15] obtained the results for ORB-SLAM2 by

[1] https://opencv.org/

running the code with default settings but turned off the loop-closure detection and global bundle adjustment. This turns ORB-SLAM2 also into a VO.

	SDSVIO				DSO [3]			
	grayscale		gradient		grayscale		gradient	
Seq.	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
00	1.700	0.266	**1.319**	**0.254**	193.260	0.292	229.790	0.266
01	**1.585**	0.105	2.946	**0.082**	19.338	0.097	13.471	0.086
02	1.546	0.208	**0.915**	**0.195**	155.910	0.220	157.300	0.221
03	2.492	**0.119**	**1.258**	0.135	15.901	0.124	13.151	0.130
04	2.656	**0.095**	**1.415**	0.159	4.377	0.199	4.501	0.160
05	2.398	0.191	**1.404**	**0.179**	96.624	0.208	102.400	0.183
06	2.873	0.189	**1.620**	**0.168**	56.320	0.201	52.944	0.173
07	4.265	0.952	**1.641**	**0.256**	45.212	0.265	48.290	0.312
08	2.109	0.251	**1.650**	**0.238**	197.790	0.273	228.890	0.262
09	1.896	0.190	**1.343**	**0.179**	32.196	0.214	42.819	0.194
10	1.036	0.175	**0.796**	**0.168**	33.132	0.165	32.487	0.182
mean	2.232	0.249	1.483	0.183	106.260	0.282	115.760	0.271

Table 5.1: Comparison of DSO vs. SDSVIO accuracy on KITTI training set, sequence $00 - 10$.
t_{rel}: translation RMSE (%)
r_{rel}: rotational RMSE [degree per 100 meters].
The best results are represented as bold numbers.



(A) Translation Error vs. Path Length

(B) Rotation Error vs. Path Length

(C) Translation Error vs. Speed

(D) Rotation Error vs. Speed

FIGURE 5.10: KITTI Average Error on Training Set (00 – 10) of SDSVIO on gradient images

	SDSVIO		St. DSO [15]		S-LSD-VO [10]		ORB-SLAM2 [24]	
Seq.	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
00	1.32	**0.25**	0.84	0.26	1.09	0.42	**0.83**	0.29
01	2.95	**0.08**	1.43	0.09	2.13	0.37	**1.38**	0.20
02	0.92	**0.20**	**0.78**	0.21	1.09	0.37	0.81	0.28
03	1.26	**0.14**	0.92	0.16	1.16	0.32	**0.71**	0.17
04	1.42	0.16	0.65	**0.15**	**0.42**	0.34	0.45	0.18
05	1.40	**0.18**	0.68	0.19	0.90	0.34	**0.64**	0.26
06	1.62	**0.17**	**0.67**	0.20	1.28	0.43	0.82	0.25
07	1.64	**0.26**	0.83	0.36	1.25	0.79	**0.78**	0.42
08	1.64	**0.24**	**0.98**	**0.25**	1.24	0.38	1.07	0.31
09	1.34	**0.18**	0.98	**0.18**	1.22	0.28	**0.82**	0.25
10	0.80	**0.17**	**0.49**	0.18	0.75	0.34	0.58	0.28
mean	1.48	0.18	0.84	0.20	1.14	0.40	0.81	0.26

TABLE 5.2: Comparison of the VO accuracy of different stereo methods on the KITTI training set, sequence 00 – 10.
t_{rel}: translation RMSE (%)
r_{rel}: rotational RMSE [degree per 100 meters].
The best results are represented as bold numbers.

Chapter 6

Discussion

This work proposes the extension of monocular VO to give a mobile robot the ability to provide motion estimation in a large-scale environment from a stereo camera supported by an inertial sensor. The unknown scale issue arising at the initialization step of a monocular approach is solved and the scale drift is eliminated. The gradient images, in combination with the proposed method, increase the accuracy on the KITTI dataset. As the results show, the extension increases the tracking accuracy.

Figure 5.2 shows the path of the trip *tunnel*. This trip is without a loop, but the vehicle drove through a tunnel. The tunnel entrance is at the 400th meter and the exit at the 900th meter. Figure 5.3 shows the absolute translation error. Here DSO falls abruptly after the tunnel entry to a much smaller scale. This is shown by the fact that the error on the z-axis increases strongly.

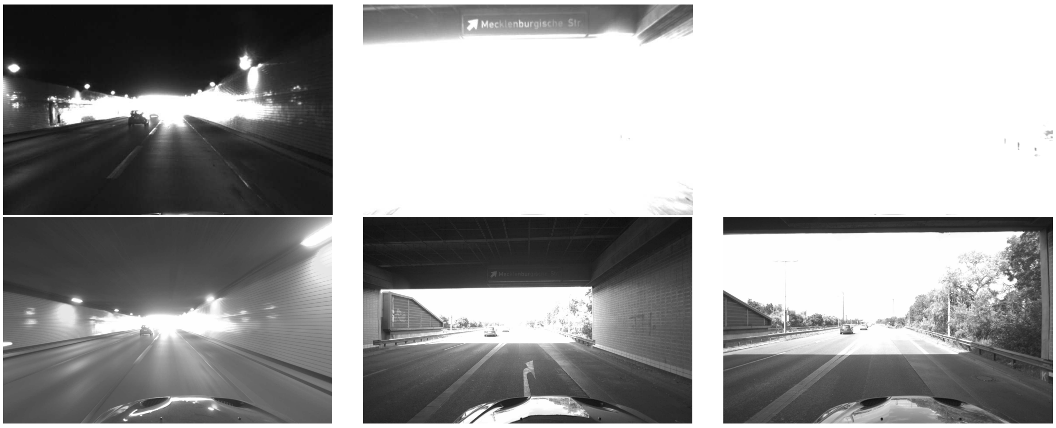


FIGURE 6.1: Three images of the stereo camera (at the top) and the corresponding grayscale images of the monocular camera (below) are shown.

The first evaluated trip *tunnel* contains the images shown in figure 6.1. The images show how challenging the images of this particular stereo camera are after a tunnel exit and not only for direct methods. Compared to the monocular camera images, the stereo images are completely overexposed. However, DSO has already big difficulties with the tunnel entrance, so that it falls into another scale. SDSVIO even manages the exit without a big accumulated drift. Nevertheless, it should not go unmentioned that the euclidean distance at the end is about 20 meters.

In the second trip *engler-loop* a loop was driven. The path plot 5.4 shows only a small difference between DSO and SDSVIO. This is also shown by the error plot 5.5. At the beginning DSO even has a lower error. Because there are no disturbing factors, e.g., other traffic participants, DSO comes through without problems.

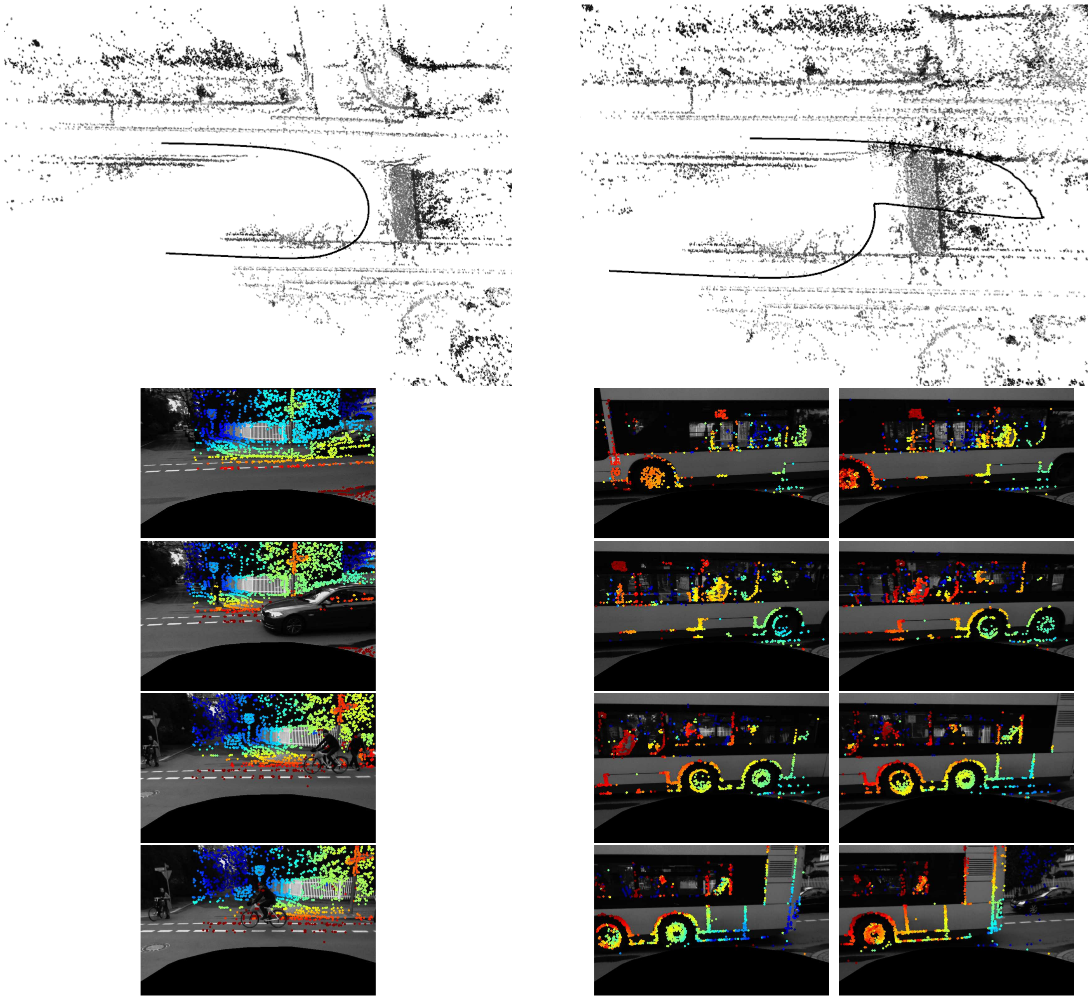


FIGURE 6.2: The left side shows the result using an inertial sensor and the right side shows the result without inertial sensor. The produced trajectory is shown at the top and the keyframes below. The colored pixels represent the tracked points and the color encode the depth values, red = near and blue = far.

The inertial sensor not only contributes to the orientation pre-initialization for new frames, but is also used for motion detection. If the inertial measurements show no movement, the tracking is not applied, therefore no keyframes are generated. Figure 6.2 shows a scene from a previous setup on the trip *engler-loop*, where the field of vision of the camera is restricted by tilting the camera down. The large black lower area masks the engine cover of the vehicle. The left side shows the result using an inertial sensor for motion detection and the right side shows the result without inertial sensor support. The trajectory is shown at the top and the keyframes below. The same scene produces more keyframes without inertial support. When the vehicle does not move and other vehicle pass it, the proposed method can be confused by passing cars. The proposed approach, without an inertial sensor, leads to incorrect results if the vehicle does not move and other vehicles in front of it do move. As the bus passes through, each frame becomes a keyframe (only every 4th is shown), so the VO is carried away. The original DSO has even been lost in the scene and has reset itself.

Figure 5.6 shows the path plot and figure 5.7 the absolute translation error of the third evaluated trip *engler-eightloop I.* On this trip DSO changes its scaling twice. DSO starts with a slightly smaller scale, probably because the vehicle does not move at the beginning. However, the scaling changes again after another stop. This increases the error of DSO compared to the error of SDSVIO.

On the fourth evaluated trip, *engler-eightloop II*, DSO starts at a slightly smaller scale and maintains this scale to the end, see figure 5.8. The average error of DSO is greater than that of SDSVIO, but the error at the end of the trip is slightly lower, see figure 5.9. This leads, apart from the wrong scale, to the conclusion that the accumulated drift of DSO is lower on this particular trip.

Another trip from a previous setup demonstrates another challenge to the proposed method. The field of vision in the previous setup is restricted by tilting the camera down. Figure 6.3 shows the focusing on a single dominant motion that leads to a wrong trajectory. This is not observed in the presented experimental setup, because now there is the opportunity that the dominant motion takes place in the high-rise buildings and trees. However, this can also happen in the presented experimental setup. This should be improved in future work.

The evaluation on the KITTI sequences in table 5.1 shows that the gradient images increase the accuracy of SDSVIO, but surprisingly decrease the accuracy in DSO. Table 5.2 shows that the combination of gradient images, which counteract the lack of photometric calibration and inertial orientation that support tracking, lead to a lower rotation error compared to other state-of-the-art methods.

The KITTI dataset provides image captures at 10 Hz while driving at a speed of up to 80 km/h. For direct methods this low frame rate is a challenge as they exploit small intra-frame movements. It is surprising that DSO tends to drift on the KITTI dataset while on our own stereo dataset it abruptly changes the scale and continues without a drift. The scale drift on the KITTI dataset may be due to the different camera FoV. However, the scale change of DSO on our own stereo dataset occurs when the environment changes, e.g., tunnel entrance or when there is no ego-motion, but new keyframes are selected. In case of SDSVIO the depth information takes care that no scale changes appear. Very small scale changes also occur in our own stereo dataset, when depth information from both stereo-view and multi-view are used and equally weighted as proposed in [10] or stereo depths are heavily weighted as proposed in [15]. This does not happen on the KITTI dataset, since the vehicle always stays in motion.

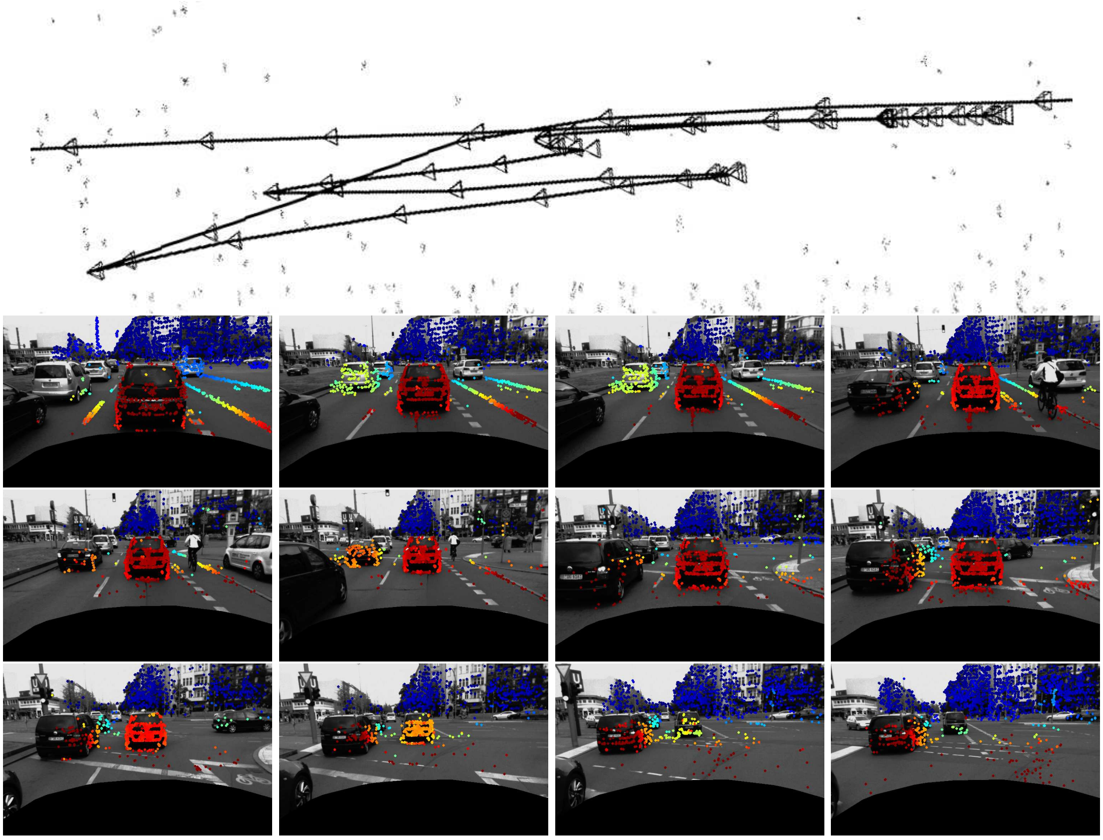


FIGURE 6.3: Focus on a single dominant motion leads to a wrong trajectory shown above. The colored pixels represent the tracked points and the color encode the depth values, red = near and blue = far.

Figure 6.4 shows a stereo and monocular camera scene. The left image from the monocular camera reveals a traffic sign in the center of the image, which is not visible on the right image of the stereo camera. The monocular camera takes into account the entire image for exposure time control. The traffic sign is not visible in the stereo images because the stereo camera controls the exposure time based on the lower half image. For this reason, especially in this scene, a stroboscopic effect caused by a repetitive environment can be observed.

Consider the upper left and lower images in figure 6.5. Even though the vehicle moved forward a line marker phase, the road surface looks almost the same, making it look like the vehicle has stopped between the upper left and lower left keyframes. The vehicle drives at a highway speed up to 90 km/h, at 30FPS which is around 0.83 meters per frame. With such speed, each frame must become a keyframe, but eleven frames are skipped because of repetitive structures. The lower left frame jumps back to the position of the upper left frame.

An advantage of a direct approach is that it also produces an accurate, semi-dense 3D point cloud containing pixels in gradient-rich areas. Figure 6.6 shows an exemplary piece of the reconstructed point cloud of the *engler-loop* trip.

For the Autonomos GmbH project, the consistency of the camera poses in relation

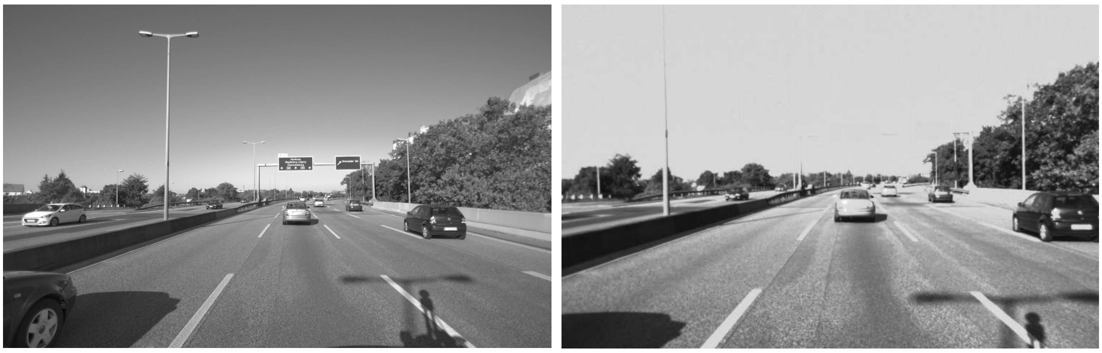


Figure 6.4: Comparison of the details of the stereo image (right) and the monocular image (left). Consider the center of the images, the left image of the monocular camera reveals a traffic sign that is not visible on the right image of the stereo camera.

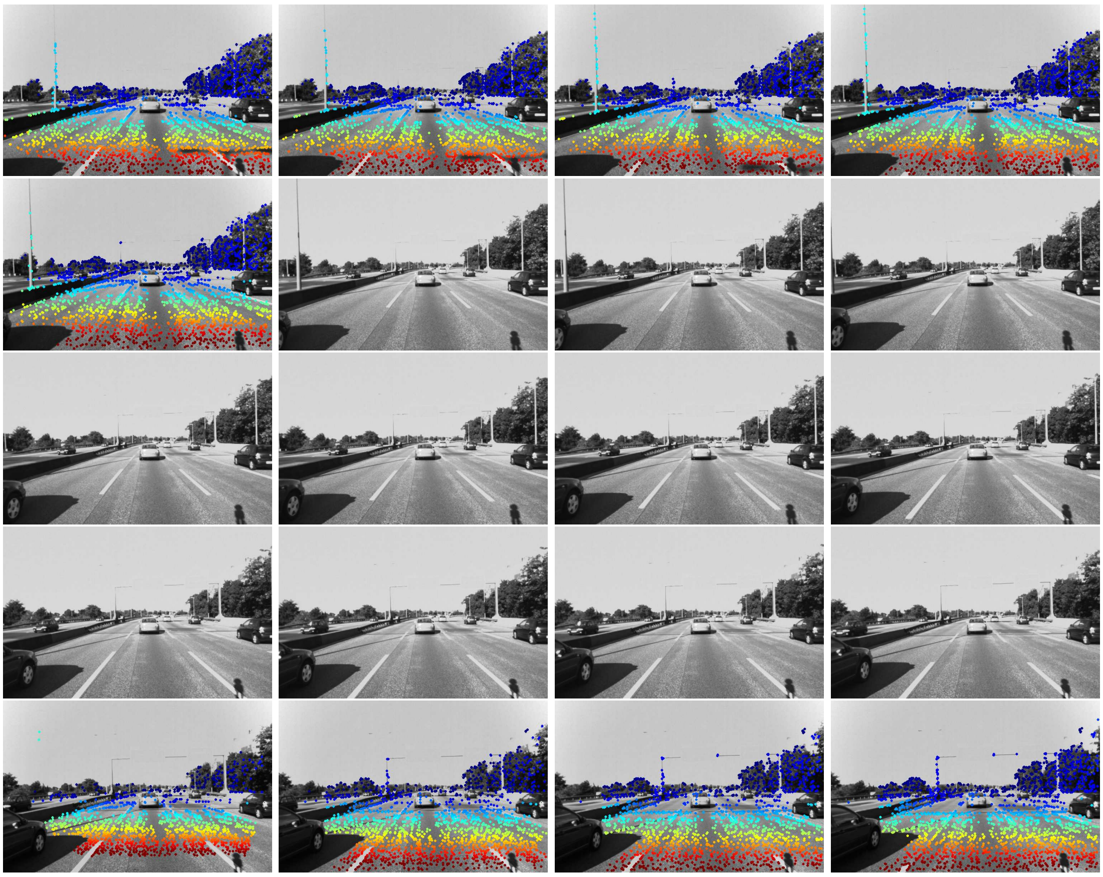


Figure 6.5: The images with colored pixels are keyframes, the others are none-keyframes. Repetitive structures cause the lower left frame to jump back to the position of the upper left frame. The colored pixels represent the tracked points and the color encode the depth values, red = near and blue = far.

to the landmarks is important. All landmarks are projected into the keyframes for the consistency checks. Figure 6.7 demonstrates ten consecutive keyframes with projected points from the entire point cloud. The yellow points indicate selected landmarks of other keyframes. Only a small subset is selected from the

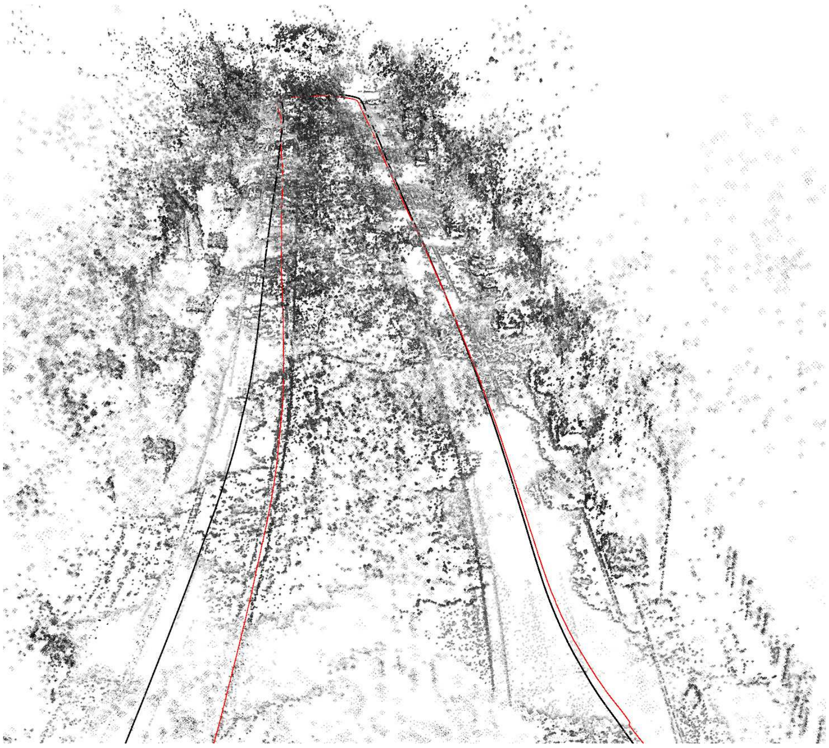


Figure 6.6: A large-scale scene reconstruction.

shown keyframes. The magenta points in a keyframe indicate selected landmarks of that keyframe. It can be seen that the points are firmly anchored in place despite a translation and rotation of the vehicle.

As the results show, the extension not only increases tracking accuracy but also robustness. Despite the fact that the extension has increases the tracking accuracy and robustness, it can still be improved. From both sides, hardware and software. Exposure control is to be improved on the camera side to provide enhanced input data. A tightly incorporation of all inertial measurements into the direct image alignment can also lead to an improvement. The frequency of the inertial sensor may need to be increased for this. Based on the outcome of this work, several tasks (e.g. ground mesh estimation and ground projection) are successfully performed despite the sometimes difficult-to-master scenes.

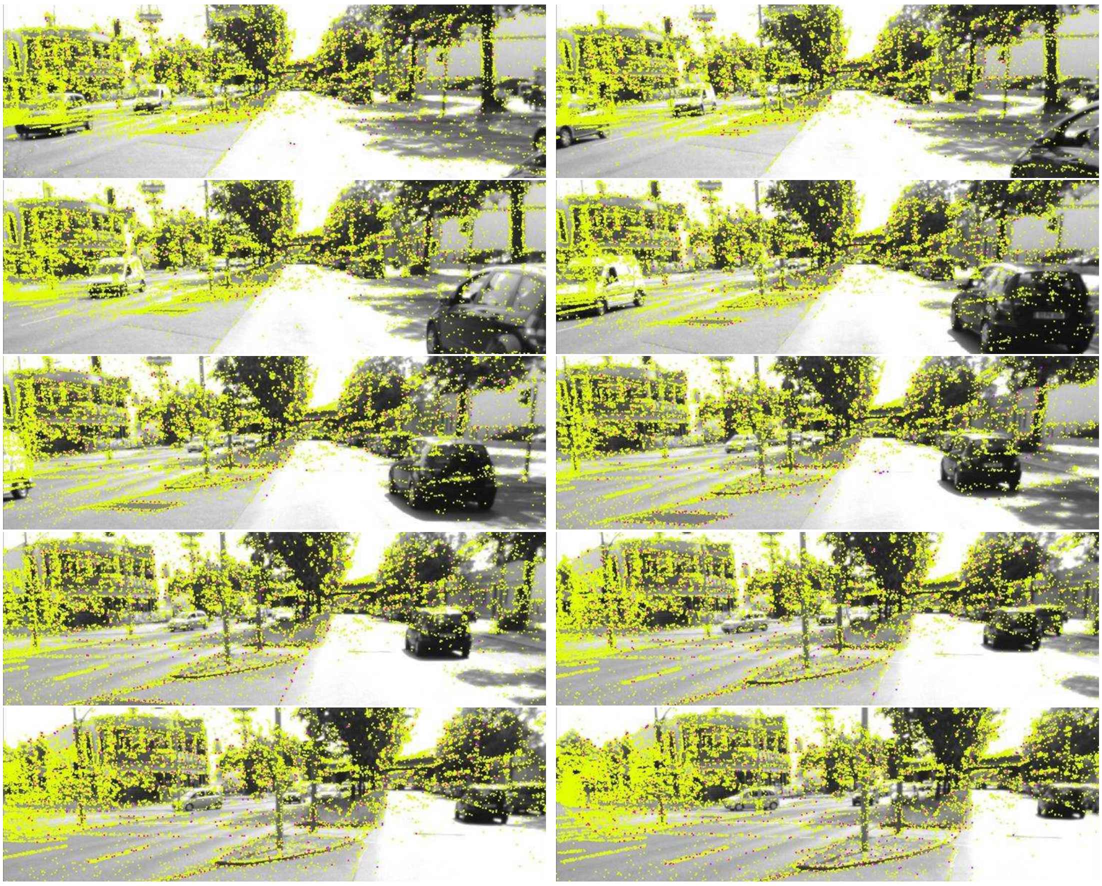


Figure 6.7: All points are projected onto keyframes and colored yellow. The hosted points of the displayed keyframes are magenta.

Chapter 7

Conclusion and Outlook

This master's thesis proposed a stereo direct sparse visual-inertial odometry. Depth information is obtained primarily from the stereo-view and secondarily, if there is no depth information from the stereo-view of the tracked pixel, from the multi-view. A semi-dense point cloud reconstruction is performed. In contrast to a previous DSO extension, this extension is able to run on pure monocular camera images (as original DSO), in addition, it can be supported in combination or individually with depth information and orientation of an inertial sensor. This allows the processing data not only from monocular and stereo cameras, but also from other sensors, e.g., RGB-D camera. As the evaluation and discussion shows, the extension has increased tracking accuracy and robustness, but can still be improved. Adjustments to the camera have to be implemented to fully exploit the strengths of the proposed method.

In future work the frequency of the inertial sensor could be increased and the inertial integration could be tightly combined with direct image alignment in the optimization. Porting the system to the stereo camera itself is also considered. A GPS integration can enhance the accuracy and robustness on the global scale, but providing absolute instead of relative poses is also interesting. Moreover, to avoid the focus on a single dominant motion, it is planned to use the, inside the Autonomos GmbH, available pixel-wise classification pipeline for masking potential dynamic objects like vehicles and persons. Nevertheless, multi rigid-body motion segmentation and estimation is more effective since it not only determines the dominant motion, but also provides motion of individual moving objects. It is also under discussion to integrate nonholonomic constraints into tracking and optimization. Finally, since the proposed method is a pure VIO and therefore suffers from accumulated drift, it could also be extended by an independent SLAM backend to enhance accuracy.

Bibliography

[1] D. Scaramuzza and F. Fraundorfer. “Visual Odometry”. In: *IEEE Robotics & Automation Magazine* (Dec. 2011), pp. 80–92.

[2] Oxford Technical Solutions Ltd. *Inertial navigation systems (INS) explained Life before GPS*. URL: `http://www.oxts.com/what-is-inertial-navigation-systems-guide/` (visited on 09/30/2017).

[3] J. Engel, V. Koltun, and D. Cremers. “Direct Sparse Odometry”. In: July 2016. arXiv: `1607.02565v2 [cs.CV]`.

[4] Hans P. Moravec. “Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover”. Ph.D. Stanford University, Sept. 1980.

[5] L. Matthies and S. Shafer. “Error modeling in stereo navigation”. In: *IEEE Journal of Robotics and Automation, Vol. RA-3, No. 3* (June 1987).

[6] D. Nister, O. Naroditsky, and J. Bergen. “Visual Odometry”. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2004).

[7] C. Kerl, J. Sturm, and D. Cremers. “Dense Visual SLAM for RGB-D Cameras”. In: *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*. 2013.

[8] J. Engel, J. Sturm, and D. Cremers. “Semi-Dense Visual Odometry for a Monocular Camera”. In: *iccv*. Sydney, Australia, Dec. 2013.

[9] J. Engel, T. Schöps, and D. Cremers. “LSD-SLAM: Large-Scale Direct Monocular SLAM”. In: *eccv*. Sept. 2014.

[10] J. Engel, J. Stueckler, and D. Cremers. “Large-Scale Direct SLAM with Stereo Cameras”. In: *International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2015.

[11] V. Usenko et al. “Direct Visual-Inertial Odometry with Stereo Cameras”. In: *International Conference on Robotics and Automation (ICRA)*. May 2016.

[12] Jianke Zhu. “Image Gradient-based Joint Direct Visual Odometry for Stereo Camera”. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, pp. 4558–4564. DOI: `10.24963/ijcai.2017/636`. URL: `https://doi.org/10.24963/ijcai.2017/636`.

[13] J. Engel, V. Koltun, and D. Cremers. “A Photometrically Calibrated Benchmark For Monocular Visual Odometry”. In: July 2016. arXiv: `1607.02555v2 [cs.CV]`.

[14] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. “ORB-SLAM: a Versatile and Accurate Monocular SLAM System”. In: *IEEE Transaction on Robotics*. 2015, pp. 1147–1163.

[15] R. Wang, M. Schwörer, and D. Cremers. “Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras”. In: International Conference on Computer Vision (ICCV), Oct. 2017.

[16] Ronald Clark et al. “VINet: Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem”. In: Apr. 2017. arXiv: `1701.08376 [cs.CV]`.

[17] Sen Wang et al. “DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks”. In: Sept. 2017. arXiv: `1709.08429v1 [cs.CV]`.

[18] N. Yang et al. “Challenges in Monocular Visual Odometry: Photometric Calibration, Motion Bias and Rolling Shutter Effect”. In: May 2017. arXiv: `1705.04300v3 [cs.CV]`.

[19] Richard Harltey and Andrew Zisserman. *Multiple View Geometry - in computer vision*. 2nd ed. Cambridge University Press, 2003. ISBN: 978-0-521-54051-3.

[20] *MPU-6000 and MPU-6050 Product Specification*. 3.2. InvenSense Inc. Nov. 2011.

[21] Stefan Leutenegger et al. “Keyframe-Based Visual-Inertial SLAM Using Nonlinear Optimization”. In: *The International Journal of Robotics Research* (2015).

[22] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

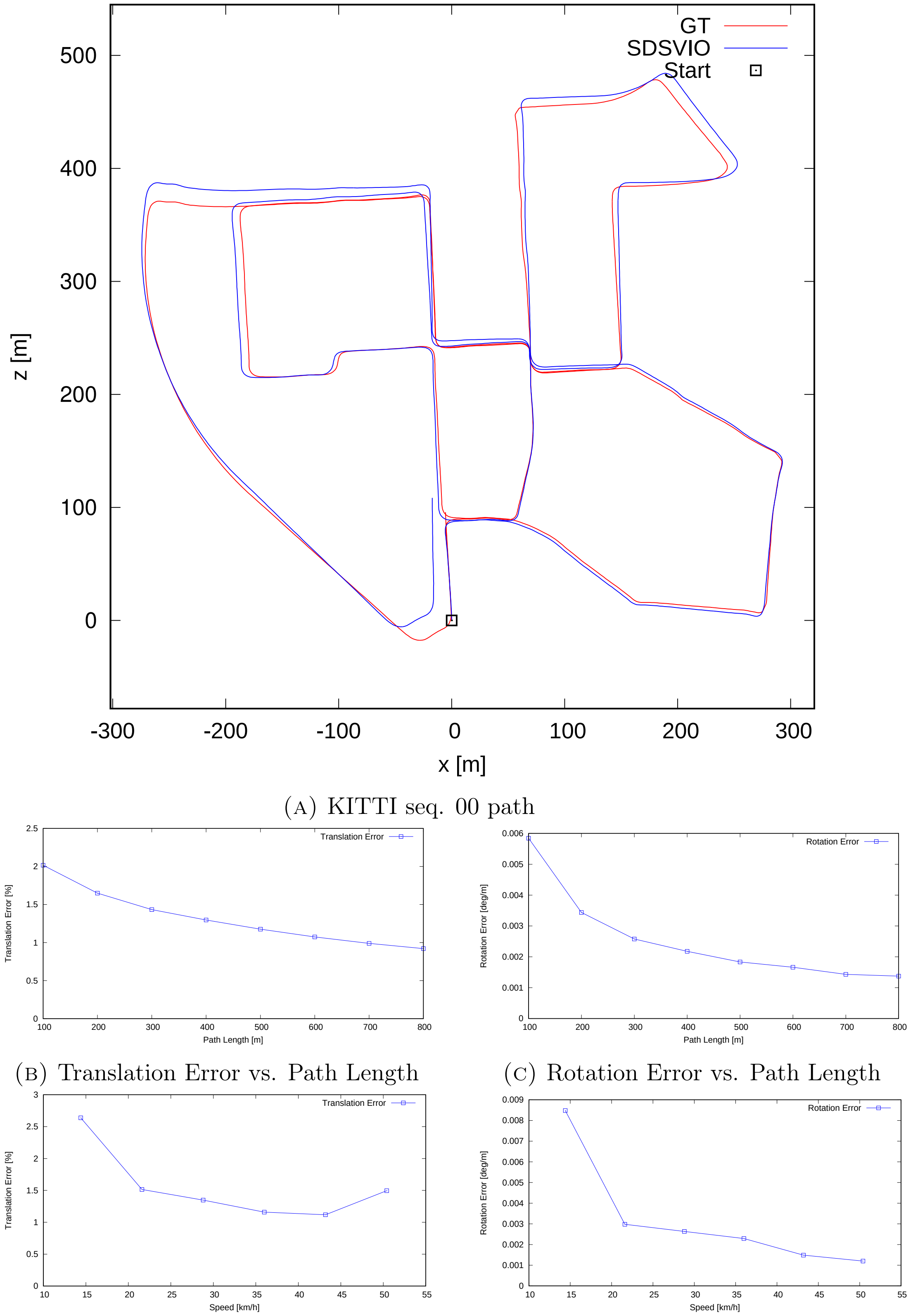
[23] Andreas Geiger et al. “Vision meets Robotics: The KITTI Dataset”. In: 2013.

[24] Raul Mur-Artal and Juan D. Tardos. “ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras”. In: June 2017. arXiv: `1610.06475v2 [cs.RO]`.

Appendix A

KITTI Evaluation

This appendix presents the full evaluation of KITTI Visual Odometry Benchmark [22] on training sequences. In all sequences IMU data was used except for sequence 03, the raw data [23] is not available for this sequence.



(A) KITTI seq. 00 path

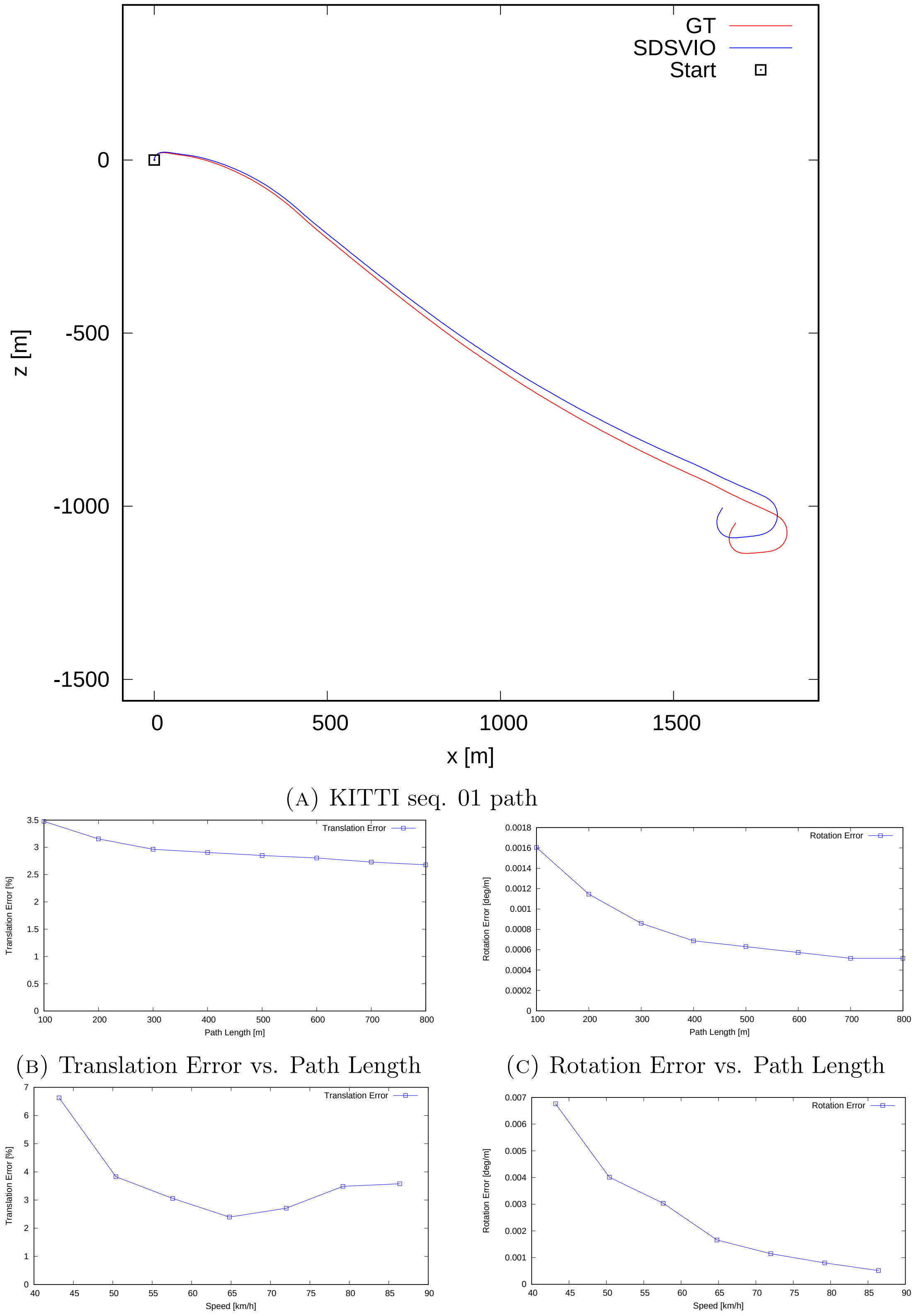
(B) Translation Error vs. Path Length

(C) Rotation Error vs. Path Length

(D) Translation Error vs. Speed

(E) Rotation Error vs. Speed

Figure A.1: KITTI seq. 00 Errors



(A) KITTI seq. 01 path

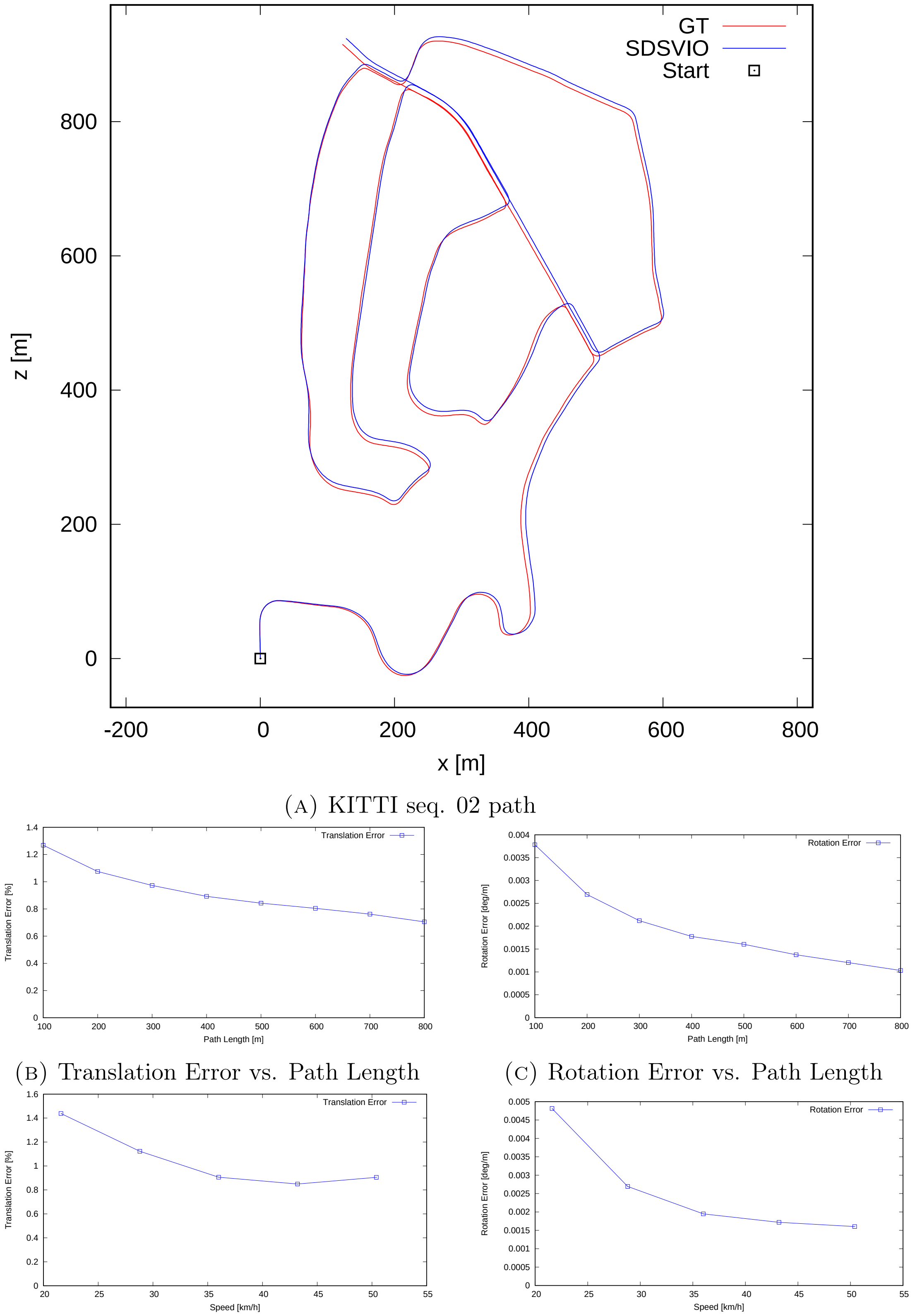
(B) Translation Error vs. Path Length

(C) Rotation Error vs. Path Length

(D) Translation Error vs. Speed

(E) Rotation Error vs. Speed

Figure A.2: KITTI seq. 01 Errors



(A) KITTI seq. 02 path

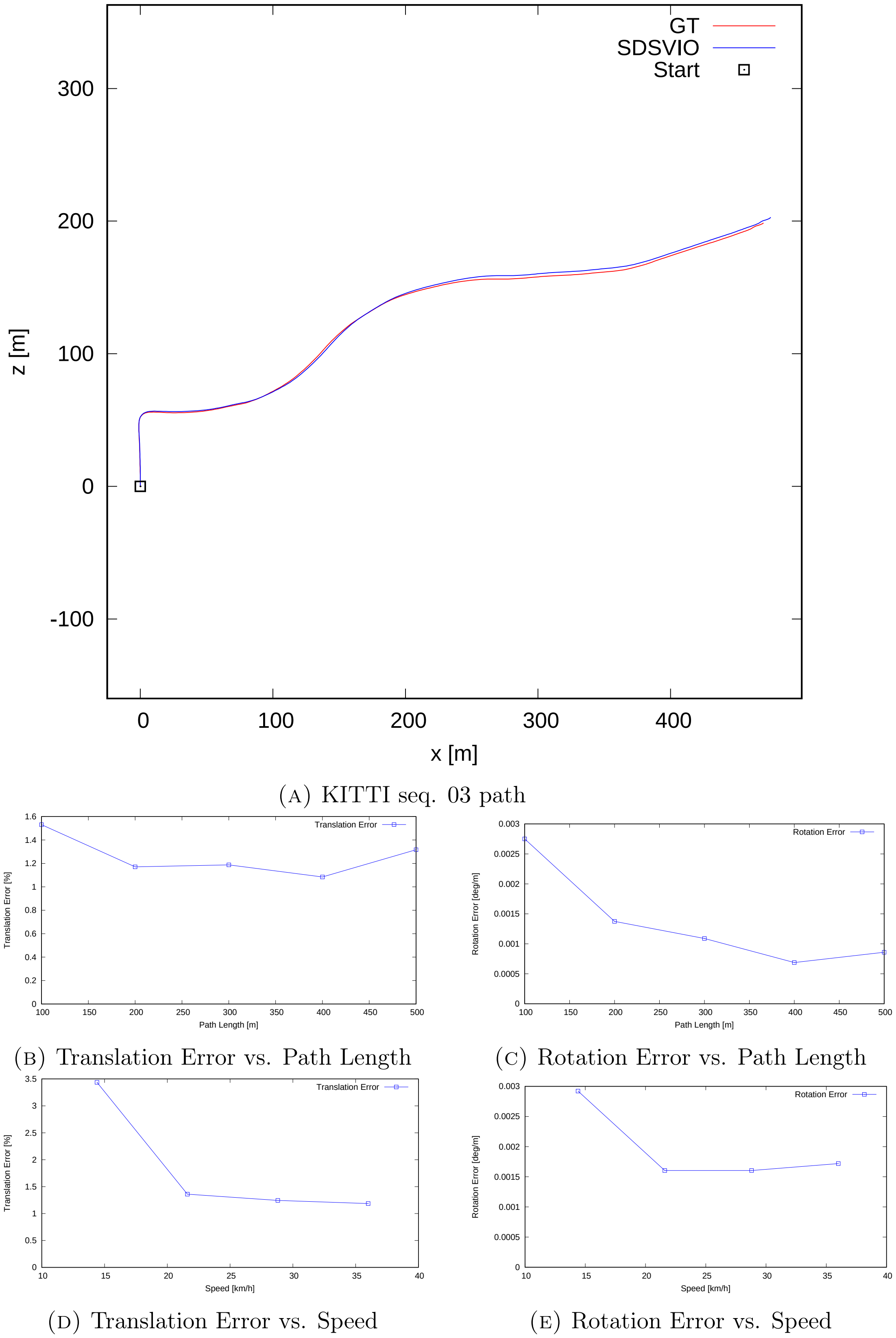
(B) Translation Error vs. Path Length

(C) Rotation Error vs. Path Length

(D) Translation Error vs. Speed

(E) Rotation Error vs. Speed

FIGURE A.3: KITTI seq. 02 Errors



(A) KITTI seq. 03 path

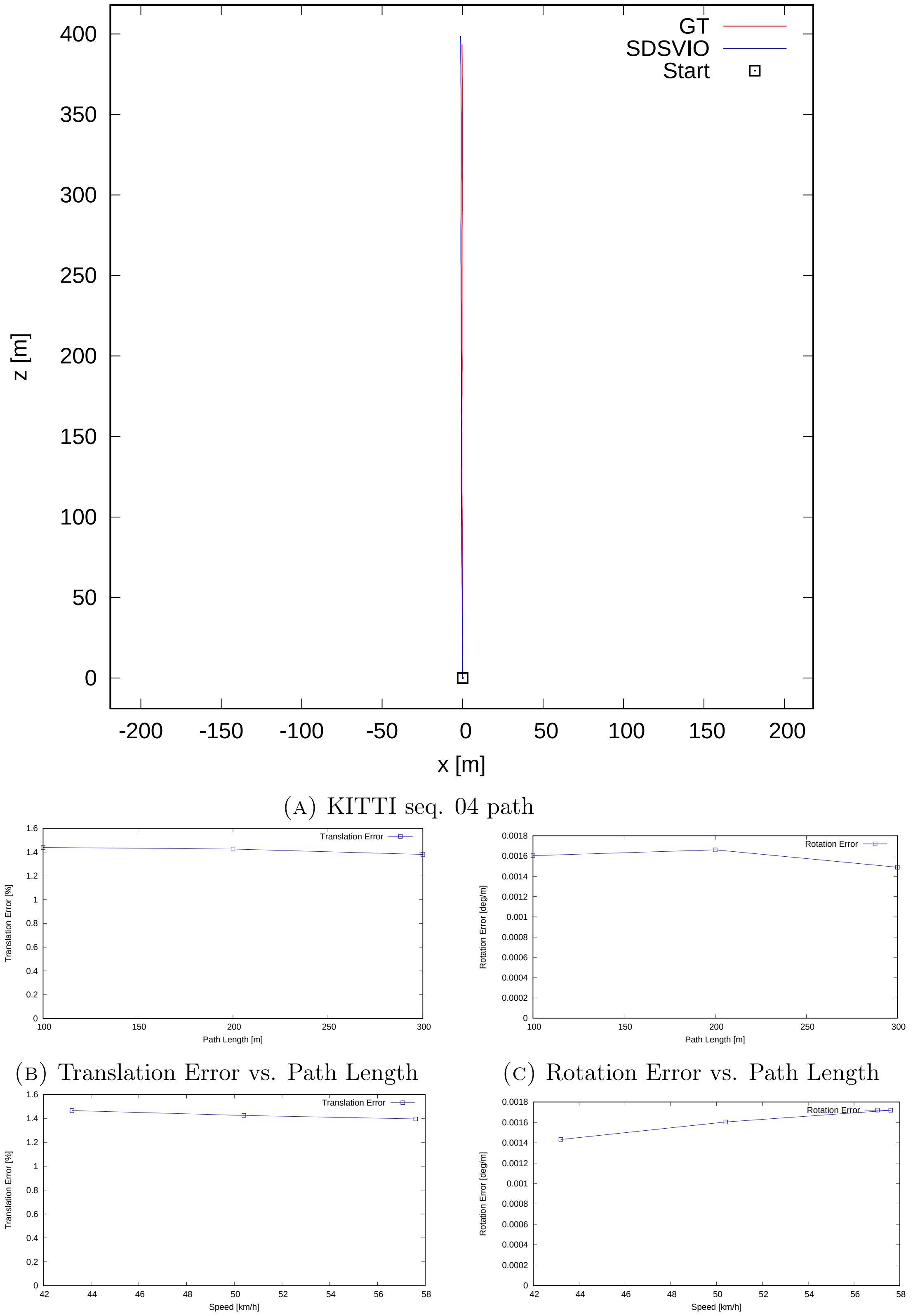
(B) Translation Error vs. Path Length

(C) Rotation Error vs. Path Length

(D) Translation Error vs. Speed

(E) Rotation Error vs. Speed

FIGURE A.4: KITTI seq. 03 Errors



(A) KITTI seq. 04 path

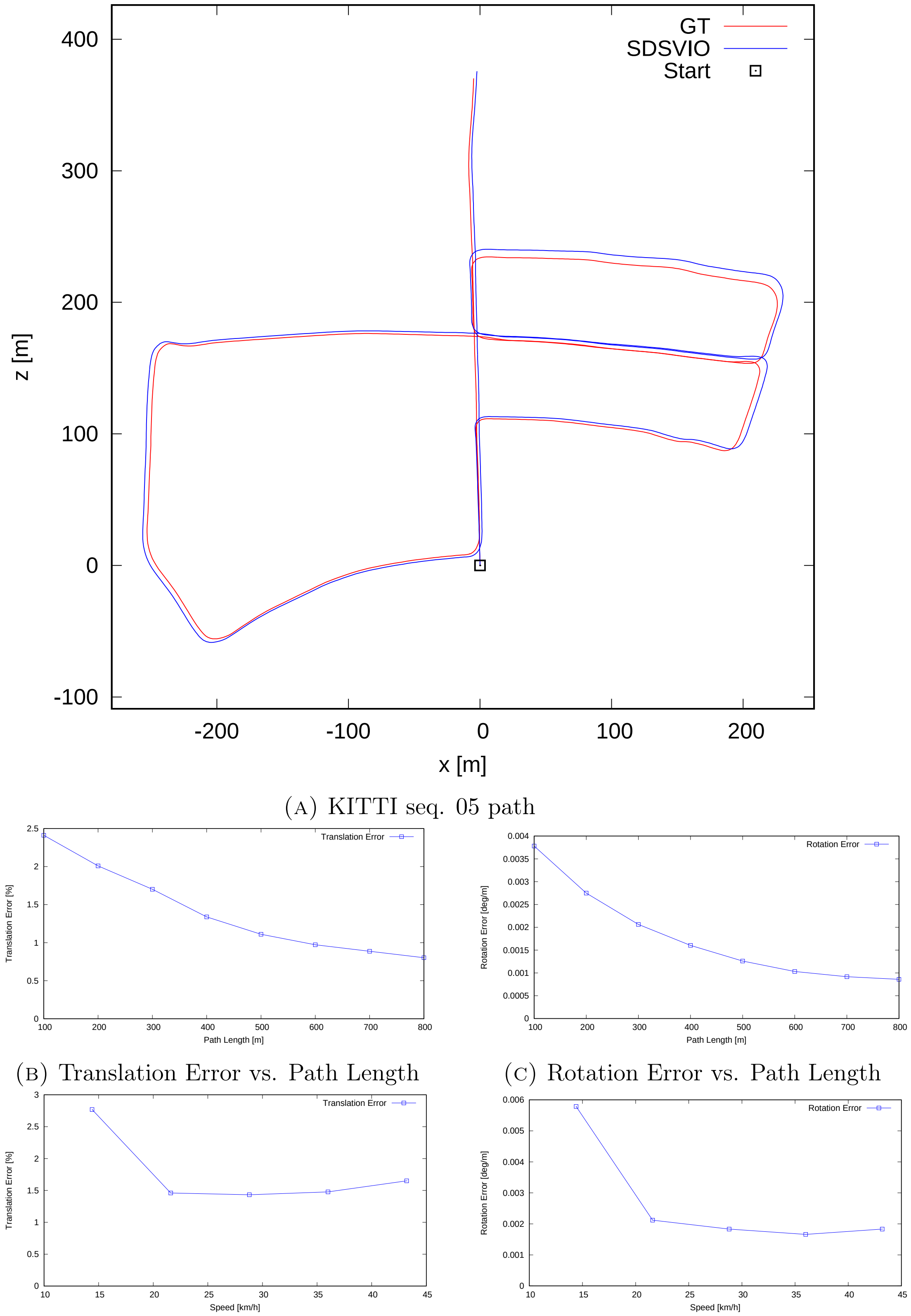
(B) Translation Error vs. Path Length

(C) Rotation Error vs. Path Length

(D) Translation Error vs. Speed

(E) Rotation Error vs. Speed

FIGURE A.5: KITTI seq. 04 Errors



(A) KITTI seq. 05 path

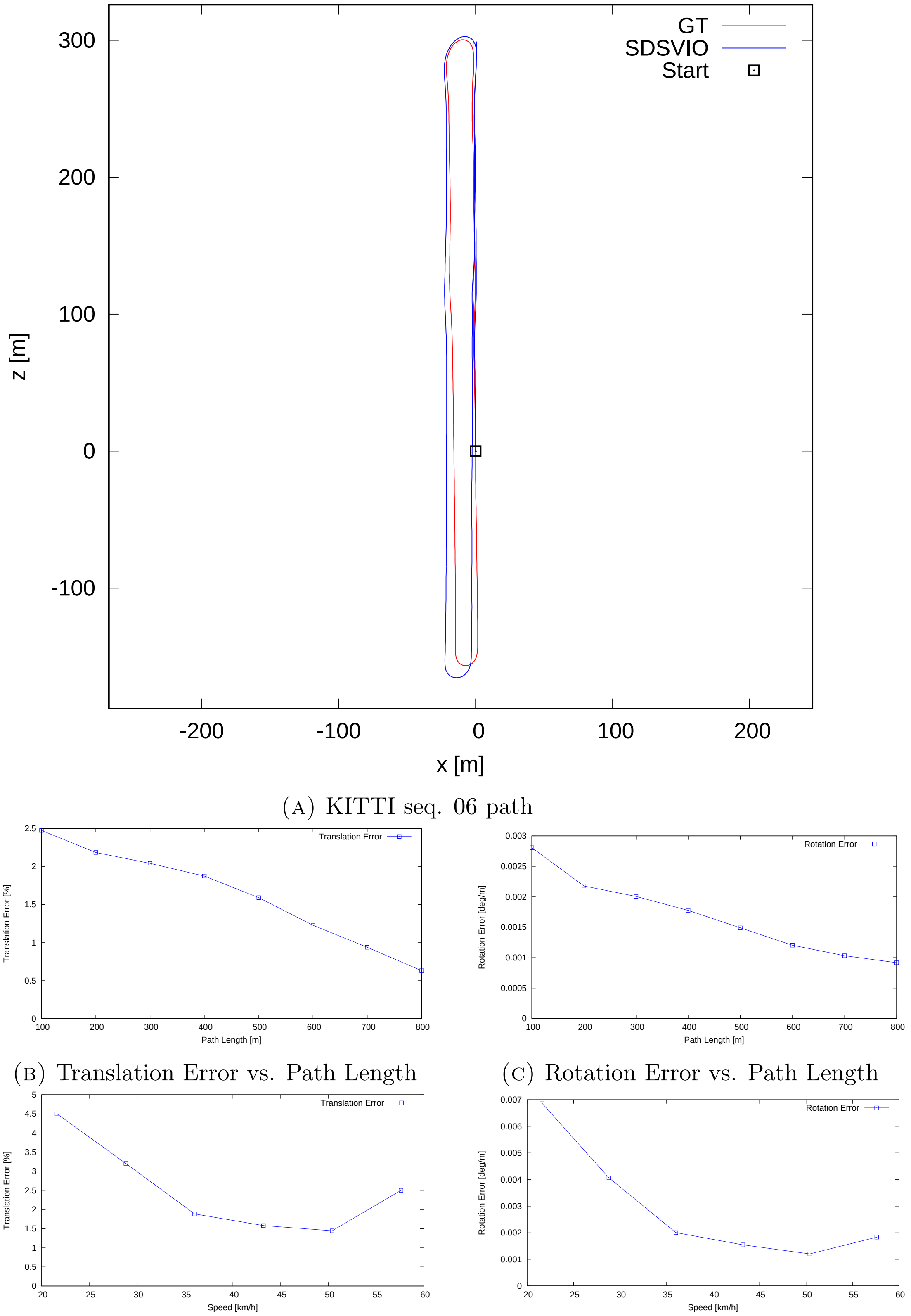
(B) Translation Error vs. Path Length

(C) Rotation Error vs. Path Length

(D) Translation Error vs. Speed

(E) Rotation Error vs. Speed

FIGURE A.6: KITTI seq. 05 Errors



(A) KITTI seq. 06 path

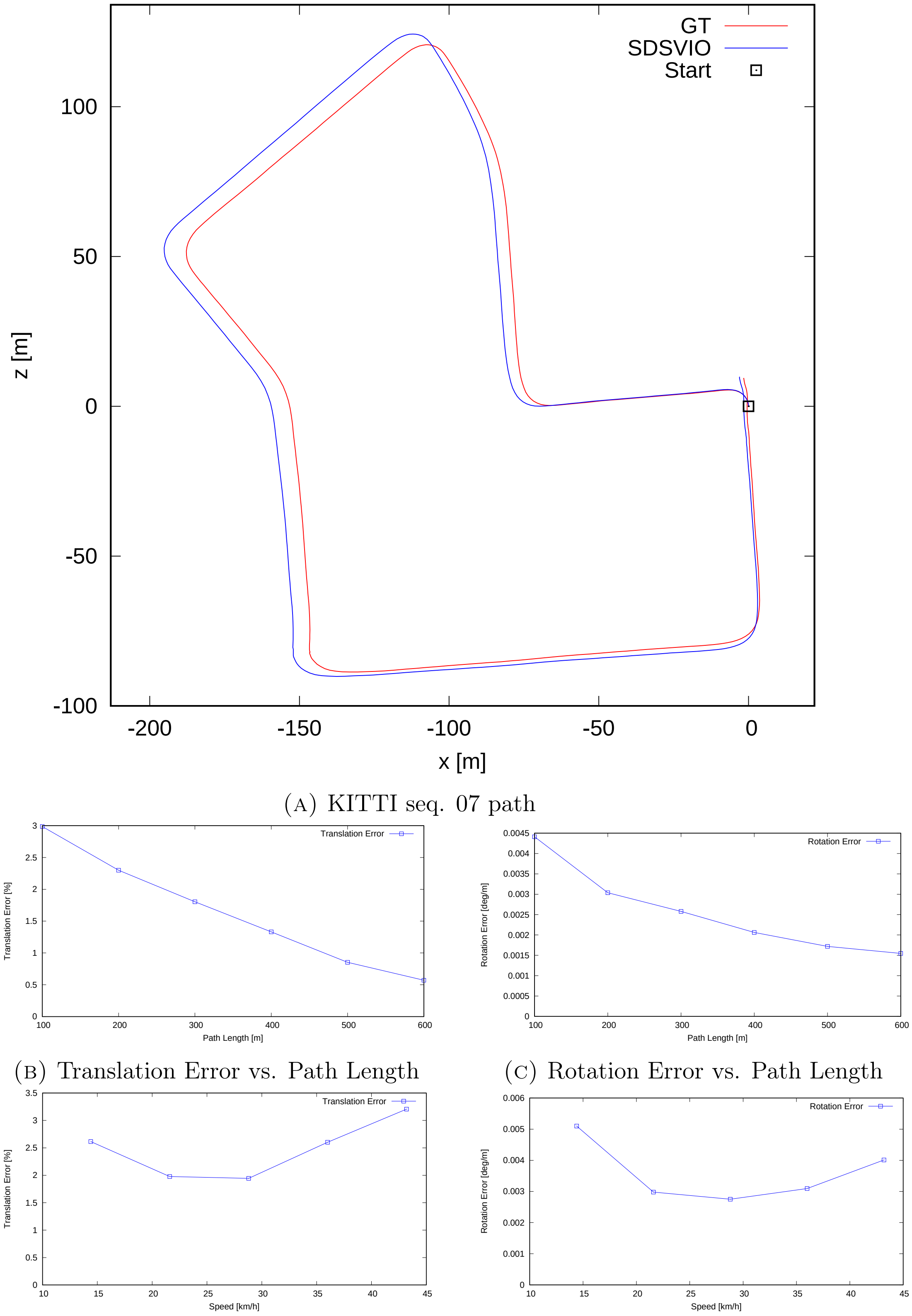
(B) Translation Error vs. Path Length

(C) Rotation Error vs. Path Length

(D) Translation Error vs. Speed

(E) Rotation Error vs. Speed

FIGURE A.7: KITTI seq. 06 Errors



(A) KITTI seq. 07 path

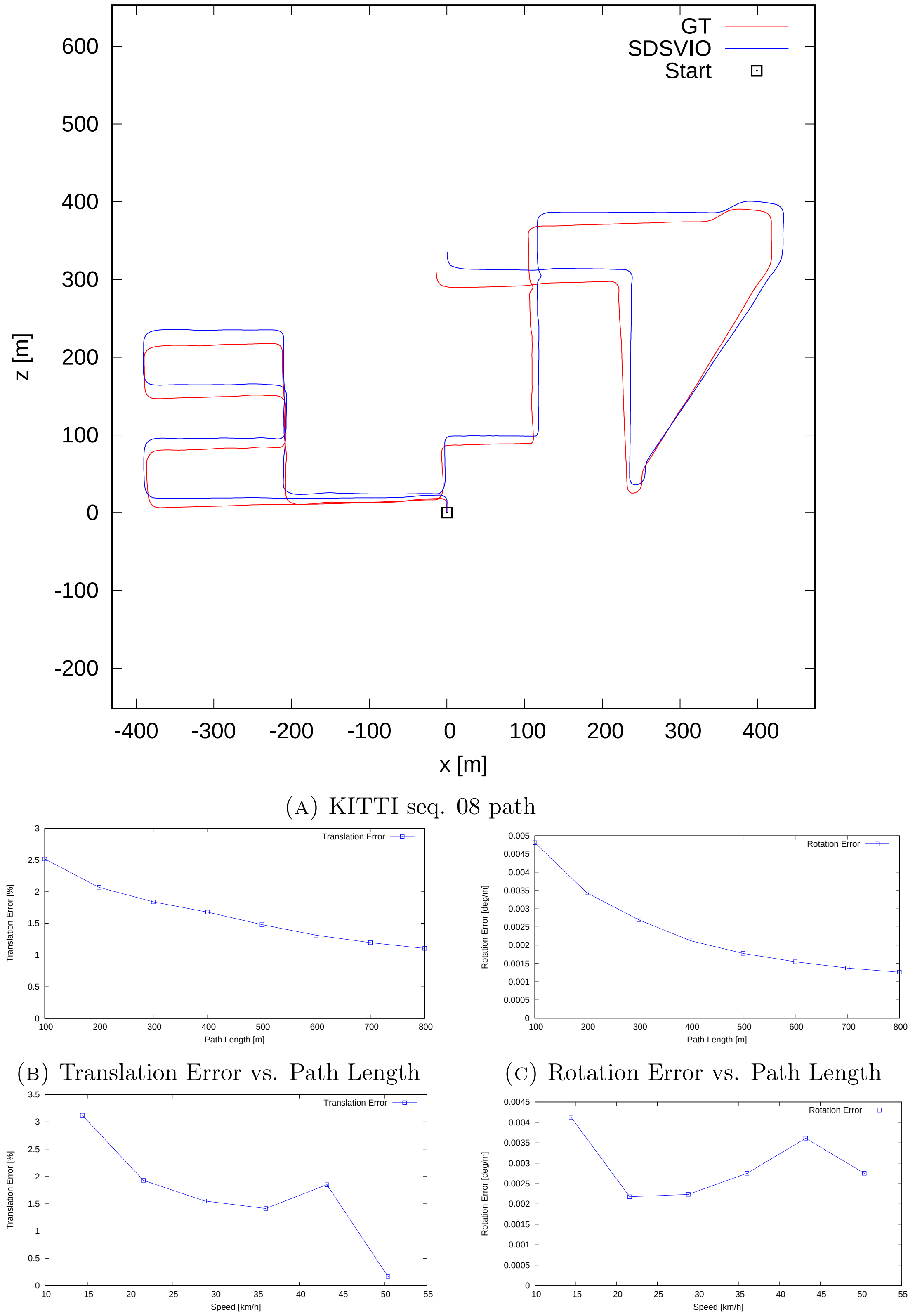
(B) Translation Error vs. Path Length

(C) Rotation Error vs. Path Length

(D) Translation Error vs. Speed

(E) Rotation Error vs. Speed

FIGURE A.8: KITTI seq. 07 Errors



(A) KITTI seq. 08 path

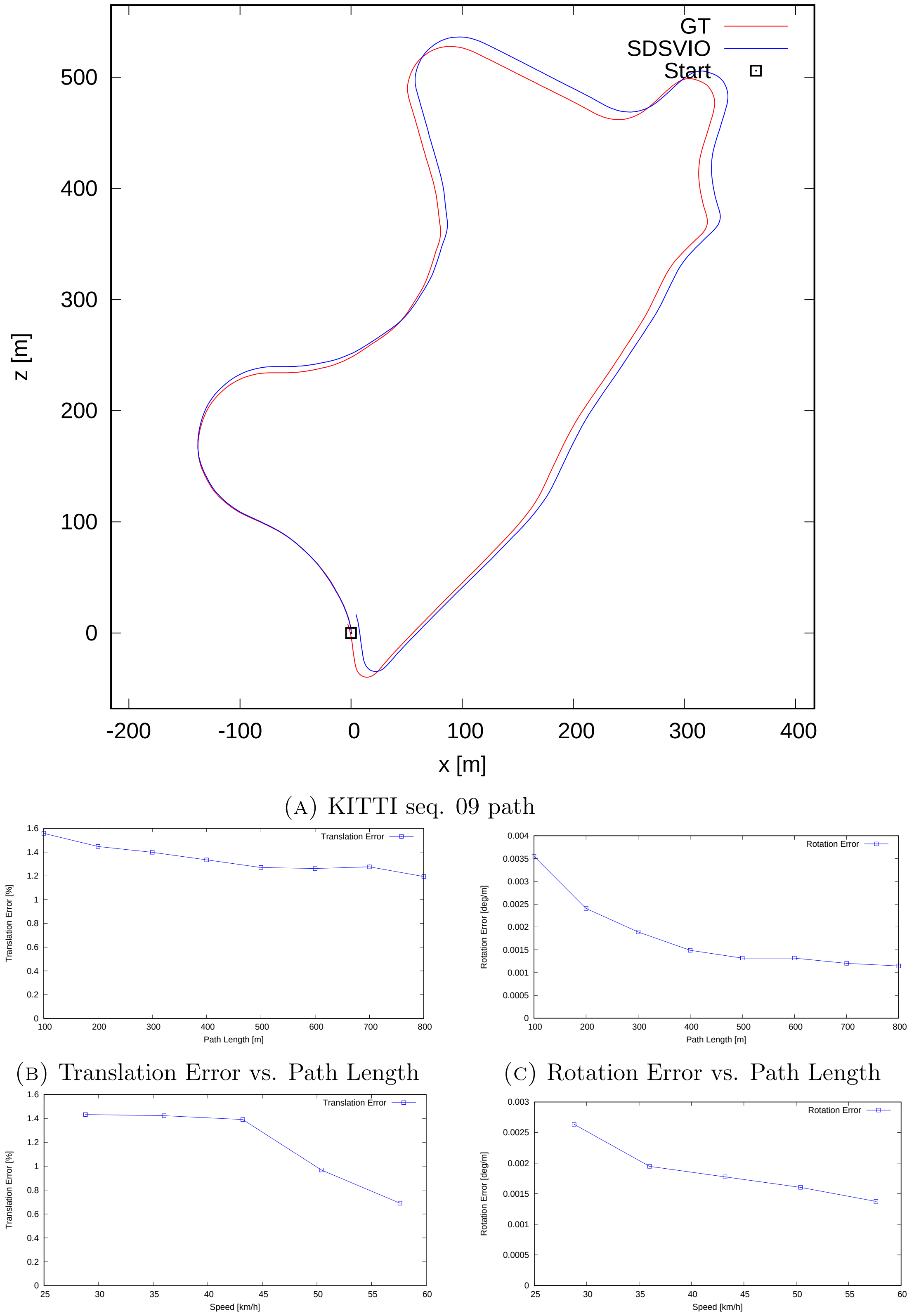
(B) Translation Error vs. Path Length

(C) Rotation Error vs. Path Length

(D) Translation Error vs. Speed

(E) Rotation Error vs. Speed

FIGURE A.9: KITTI seq. 08 Errors



(A) KITTI seq. 09 path

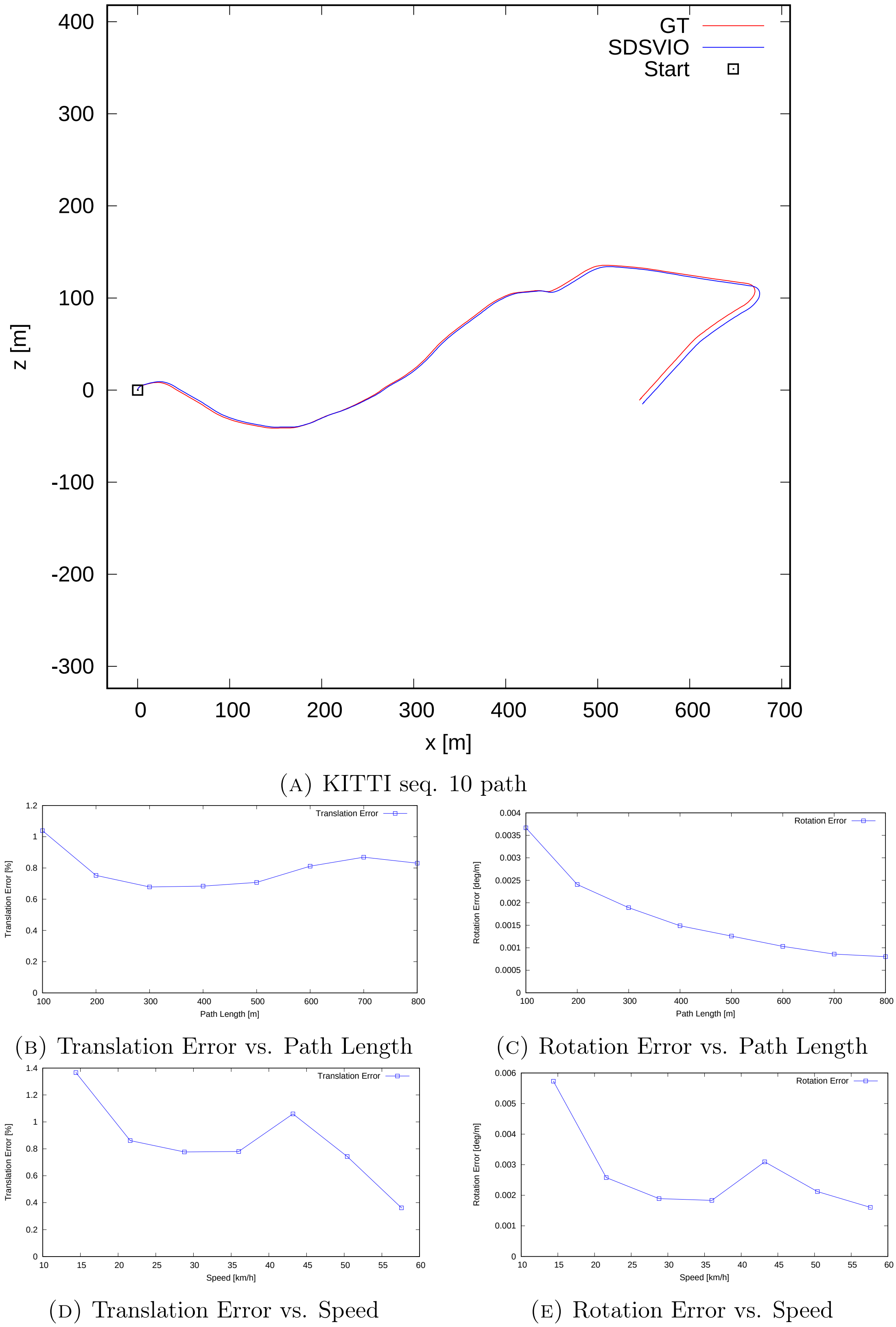
(B) Translation Error vs. Path Length

(C) Rotation Error vs. Path Length

(D) Translation Error vs. Speed

(E) Rotation Error vs. Speed

FIGURE A.10: KITTI seq. 09 Errors



(A) KITTI seq. 10 path

(B) Translation Error vs. Path Length

(C) Rotation Error vs. Path Length

(D) Translation Error vs. Speed

(E) Rotation Error vs. Speed

FIGURE A.11: KITTI seq. 10 Errors