UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

FAKULTÄT FÜR
ELEKTROTECHNIK,
INFORMATIK UND
MATHEMATIK

Robust Techniques for Monocular Simultaneous Localization and Mapping

Zur Erlangung des akademischen Grades

DOKTORINGENIEUR (Dr.-Ing.)

der Fakultät für Elektrotechnik, Informatik und Mathematik
der Universität Paderborn
vorgelegte Dissertation
von

MSc-EE. Mohammad Hossein Mirabdollah

Paderborn

Referentin: Prof. Dr.-Ing. Bärbel Mertsching
Korreferent: Prof. Dr.-Ing. Reinhold Häb-Umbach

Tag der mündlichen Prüfung: 04.02.2016
Paderborn, den 30.03.2016
Diss. EIM-E/323

Dedication

Dedicated to my wife Vahideh who went through all the hardships during the time of this research very patiently.

Declaration

I hereby declare that I have completed the work on this PhD dissertation with my own efforts and no part of this work or documentation has been copied from any other source. It is also assured that this work is not submitted to any other institution for award of any degree or certificate.

Paderborn, March 30, 2016

Mohammad Hossein Mirabdollah

Kurzfassung

Als SLAM-Verfahren (engl. Simultaneous Localization and Mapping) werden Algorithmen bezeichnet, die eine zuverlässige Schätzung der Pose eines beweglichen Fahrzeugs (Roboter) innerhalb einer Umgebung vornehmen und gleichzeitig eine geometrische Karte der Umgebung erstellen, durch die sich das Fahrzeug bewegt. Um eine grobe Schätzung der Pose eines Fahrzeugs vorzunehmen, werden gewöhnlich Sensoren wie Gyroskope, Radencoder oder visuelle Odometrie-Systeme eingesetzt. Die genannten Sensoren liefern Informationen über die momentane Bewegung des Fahrzeugs, die unter Verwendung geeigneter Bewegungsmodelle zur Schätzung der Pose des Fahrzeugs in jedem Zeitschritt herangezogen werden können. Die damit erhaltenen Bewegungsinformationen sind jedoch ungenau, wodurch sich die Fehler bezüglich der Schätzung der Pose des Fahrzeugs akkumulieren. Um dieser Herausforderung zu begegnen, wurden daher SLAM-Verfahren entwickelt, die extrinsische Sensoren zur Durchführung zusätzlicher Messungen verwenden und diese mit weiteren Daten verknüpfen. Dadurch kann die Fehlerakkumulation eingeschränkt und eine genaue Karte der Umgebung erstellt werden. Traditionelle SLAM-Verfahren verwenden extrinsische Sensoren, die Abstands- und Winkelmessungen zwischen dem Fahrzeug und speziellen Merkmalen der Umgebung (Landmarken) liefern. Zu diesem Zweck werden in der Regel Laser-Entfernungsmesser oder Stereokameras eingesetzt. Laser-Entfernungsmesser sind jedoch im Allgemeinen sehr teuer und hauptsächlich für Innenraumumgebungen geeignet. Die Verwendung von Stereokameras erfordert aufgrund der Assoziation von Merkmalen zwischen den beiden Kameras zusätzliche Rechenkapazitäten. Befinden sich die Merkmale weiterhin in einem großen Abstand zur Stereokamera, weisen die geschätzten Positionen der Landmarken hohe Ungenauigkeiten auf, die sich nicht mehr durch gaußsche Modelle beschreiben lassen. Daher ist in den letzten Jahren die Verwendung monokularer

Kameras in den Fokus der Forscher gerückt. Entsprechende SLAM-Verfahren werden in der Literatur als *Monocular SLAM* bezeichnet und bestehen aus drei Teilen: Erstens der Exktraktion und Verfolgung von Merkmalen zwischen Einzelbildern, zweitens der visuellen Odometrie und drittens der Fusion der Merkmalsverfolgung mit Odometriedaten.

Im Rahmen dieser Arbeit untersuchen wir zunächst verschiedene Verfahren zur Verfolgung optischer Merkmale innerhalb aufeinanderfolgender Einzelbilder. Dabei wird gezeigt, dass der als *Lucas-Kanade Sparse Optical Flow* bekannte Ansatz in einem Vorwärts-Rückwärts-Schema im Vergleich zu anderen Arten von Verfolgungsalgorithmen erheblich besser bezüglich der Genauigkeit und Berechnungszeit ist.

Weiterhin liefern wir Beiträge zur monokularen visuellen Odometrie, indem wir im Vergleich zu existierenden Ansätzen robustere Methoden gegen Messrauschen (8-Punkt-, 7-Punkt- und 5-Punkt-Methoden) entwickeln. Außerdem wird eine robuste Methode vorgestellt, die optische Merkmale von niedriger Qualität auf sehr homogenen Grundflächen verfolgen kann. Damit lässt sich auch die Skalierung der Bewegung schätzen, wenn nur eine monokulare Kamera mit bekannter Höhe auf einem beweglichen Fahrzeug mit Rädern zur Verfügung steht.

Schließlich stellen wir neue nicht-gaußsche Fusionsansätze zum Umgang mit hohen Ungenauigkeiten von Tiefeninformationen vor. Bei diesen neuen Methoden werden die Unsicherheiten bezüglich der Landmarken mit Hilfe von Liniensegmenten modelliert, die auf Basis einer probabilistischen Triangulation angepasst werden. Die Pose des Roboters wird in jedem Zeitschritt auf Basis von Partikelfiltern oder einer *Subspace-Technik* angepasst.

Wir evaluieren die vorgestellten Methoden für monokulare visuelle Odometrie und SLAM anhand von anspruchsvollen simulierten und realen Datensätzen und zeigen signifikante Verbesserungen im Vergleich zu existierenden State-of-the-Art-Methoden auf.

Abstract

Simultaneous Localization and Mapping (SLAM) refers to the algorithms utilized to achieve a reliable estimation of the pose of a mobile vehicle (robot) and also a geometric map of the environment through which the vehicle moves. To have a rough estimation of a vehicle's pose, usually, sensors such as gyroscopes, wheel-encoders or visual odometry systems are applied. The sensors provide instantaneous motion parameters of the vehicle which can be used along with proper motion models to estimate the pose of the vehicle at each time instance. Nevertheless, the mentioned sensors cannot provide precise motion parameters, giving rise to accumulating errors concerning the vehicle's pose. To address this problem, SLAM algorithms have been introduced to utilize extrinsic sensors to obtain extra measurements and to fuse those with other sources of data in order to limit the increasing errors and generate a reliable map. In conventional SLAM systems, extrinsic sensors are used, which deliver distance and bearing measurements between the vehicle and special features (landmarks) in an environment. The sensors for such measurements could be laser range finders or stereo cameras. Laser range finders are generally very expensive and they are mostly suited for indoor environments. Using stereo cameras necessitates extra computation loads for feature associations between two cameras and if the features are far from the camera (depending on the resolution of the camera), the estimated positions of the landmarks have large uncertainties which cannot be modeled by Gaussian models. Therefore, applying monocular cameras has attracted the attentions of researchers in the recent years. This type of SLAM is known in literature as monocular SLAM which consists of three parts. First, feature extraction and tracking within the frames. Second, visual odometry and third, the fusion of the feature tracking information with odometry data.

In this work, we firstly investigate different techniques for optical feature tracking within consecutive frames and show that the Lucas-Kanade sparse optical flow method in a forward-backward scheme outperforms other types of trackers to a great extent both in the senses of precision and computation time.

Additionally, we contribute to monocular visual odometry by developing methods which are more robust against measurement noise in comparison to the existing methods (8-point, 7-point and 5-point methods). Furthermore, a robust technique is proposed to track low quality optical features on highly homogeneous ground planes in order to estimate motion scale if the only source of data is a single camera installed on a wheeled vehicle at a known height.

Finally, we introduce new non-Gaussian fusion methods to cope with high depth uncertainties in the monocular SLAM problem. In these new methods, the uncertainties of landmarks are modeled by line segments and the line segments are trimmed based on a probabilistic triangulation. The robot pose is modified at each time instance based on particle filters or a subspace technique.

We evaluate the proposed methods concerning monocular visual odometry and SLAM based on demanding simulated and real datasets and demonstrate the significant improvements in comparison to existing state-of-the-art methods.

Acknowledgements

First of all, praise and thanks be to God who enabled me to reach this level of academic achievement. Secondly, I am grateful to my parents and wife whose moral support and prayers made it come so far.

I am extremely thankful to my supervisor Professor Bärbel Mertsching whose guidance and support in all helpful aspects during the time of my PhD kept things advancing.

Furthermore, I am grateful to my colleagues Dirk Fischer and Markus Hennig for their technical supports.

Contents

1 Introduction

Visual simultaneous localization and mapping (vSLAM) stands for the approaches which use camera data to reduce the errors of a basic navigation system of a mobile robot (vehicle) and generate a map of visual features.

Basic navigation systems of a mobile robot utilizes wheel encoders (for planar motions), IMU (inertial measurement unit) sensors or visual odometry systems to obtain the instantaneous motion parameters of the robot. The pose of a robot at each time instance can be obtained by applying the motion parameters into the motion model of the robot. Nevertheless, due to various disturbances, the sensors cannot provide the motion parameters precisely. This imprecision may occur continuously over time, resulting in an accumulating error concerning the robot pose. To tackle this problem, extra sensors can be used which provide relative measurements (distances or bearings) between the robot and some landmarks in an environment. The new measurements can be fused with the odometry data to obtain a modified version of the robot pose. This approach is known in literature as simultaneous localization and mapping (SLAM), in which the absolute robot and landmark positions are concurrently estimated. In common SLAM approaches, the relative distances and bearings between the robot and the landmarks are used. The typical sensors for these measurements are laser range finders or stereo cameras. Laser range finders are typically very expensive and have limited measurement ranges; additionally, the types and number of features which can be extracted from their data are limited. Using stereo cameras requires extra computational loads to match features in the right and the left camera images and for the relatively far landmarks they cannot provide proper depth estimations. Therefore, using monocular cameras for the SLAM problem has become popular in the recent years. Using a monocular camera has, however, the disadvantage that the relative distances to the landmarks are missing and

it can work only as a bearing measurement sensor. Therefore, more complex algorithms should be applied to compensate this shortcoming.

Generally, implementations of monocular visual odometry and SLAM algorithms consist of three main parts: (i) feature extraction and association, (ii) estimation of instantaneous motion parameters and (iii) fusion. Regarding these parts, this chapter introduces related works for optical feature tracking in section 1.1, visual odometry in 1.2 and fusion methods in 1.3. In section 1.4, the contribution of this work to monocular visual odometry and the SLAM problem is discussed.

1.1 Visual Feature Extraction and Tracking

To use a camera as a measurement device, we need to extract features in one frame and track them in the next frames. From an image, generally, three types of low level features can be extracted: edges, corners and blobs. Edge tracking has been investigated in some limited works, since it is usually computationally expensive and additionally by changing scale of objects the edges cannot be precisely localized. Therefore, in practical applications, tracking of corners or blobs which are known as point features are much more interesting.

The point feature tracking problem has been addressed in literature from two different points of view. In the first approach (called image matching), point features (keypoints) are extracted from two images separately. Then, based on surrounding points of the features, a descriptor vector for each keypoint is defined. Finally, by comparing the descriptors of the keypoints from two images, the best matching points are found. Several works were done concerning this approach, of which the more robust and reliable are the scale invariant feature transform (SIFT) [Low04] and speed up robust features (SURF) [HTG08].

Using difference of Gaussians at different scale and octaves, SIFT extracts blobs with different scale (size) as keypoints. The SIFT descriptors are found by analyzing different regions about the neighborhood of a keypoint and building up a histogram of intensity gradients based on the intensities in the regions. The length of a SIFT descriptor vector is typically 128. Although SIFT is known to have a low number of wrong feature associations (outlier rate), which was proven in different applications, it suffers from high computation time making

it inappropriate for real time applications. To address the slow speed of SIFT, the SURF algorithm was proposed, in which two simple masks are used to extract interest points. To find features at different scale, SURF changes the size of the masks instead of scaling and resizing images. The descriptors are also found using a fast convolution with first order Haar wavelets, which results in a descriptor vector with the length of 64. The mentioned modifications have made SURF much faster than SIFT, however, its inlier ratio is not as good as SIFT [MM12a].

Image matching methods provide mostly a high ratio of inliers. Nevertheless, they are sensitive to the repeatability of features in an image sequence. In other words, if we want to track a point from one frame in the next frames, it is necessary that the point is already detected as a keypoint in the next frames; otherwise, there is not any chance to track the point. Therefore, usually it is not possible to track a feature using SIFT or SURF in more than a few frames.

The second approach to address the feature tracking problem is the use of the optical flow. Optical flow methods attempt to estimate the motion of each point between two frames. The theory of optical flow is not based on any features. However, since in many applications the calculation of the motion of all points are not required, motion of feature points are calculated.

There are three types of optical flow methods: differential methods, block matching methods and methods based on the frequency domain. Differential methods solve a spatiotemporal differential equation by assuming that the intensity of a point does not change between two frames. In the basic equation of the differential optical flow method, there are two unknown parameters and therefore another constraint must be used to determine the parameters uniquely. Depending on the constraints, various differential methods have been proposed [HS81], [LK81], [UGVT88].

If the frames are noisy or they have low resolutions, the spatiotemporal differentiation cannot provide reliable information about the point displacements. As a result, the idea of block matching between frames emerged [Ana89], [Sin90]. Nevertheless, these methods are not accurate and are not appropriate for visual odometry purposes.

Another group of methods approach the optical flow problem in the frequency domain by applying velocity filters in spatiotemporal space and using the output responses of the filters. To this end, different filters each of which is sensitive to a direction of motion are utilized. The magnitudes of the outputs of filters are used in [Hee87] and the filter which gives the maximum magnitude determines the velocity of a point. In [FJ90], instead of magnitude information, phase information is used. The authors discuss that phase information is less affected by noise than magnitude information. The performance of the methods in the frequency domain, however, depends on the number of filters. In case of large optical flows, the number of filters increase dramatically, which make the methods useless for the calculation of large optical flows.

In this work, we will evaluate feature matching techniques and a sparse optical flow method for tracking of features in a set of consecutive frames. We will show that if the sparse optical flow is used in a forward and backward scheme, a fast reliable tracker is obtained, which outperforms the other methods to a great extent.

1.2 Visual Odometry

As already discussed, SLAM algorithms require to be fed by a rough estimation of motion parameters of a robot and then utilize relative measurements between the robot and landmarks from an environment to enhance the estimation of the robot pose. A well-known sensor for the motion measurement is an IMU (Inertial Measurement Unit) sensor which can provide a rough estimation of the motion parameters with respect to a global coordinate frame attached to the earth. In 3D cases, a mobile robot can have six degrees of freedom: three angular and three translational velocities with respect to the three axis of the world frame (X, Y and Z). In the case of planar movements, a robot pose can be represented by three parameters and its motion can be determined by two translation and one rotation parameters. For a wheeled robot the motion could be more constrained and determined with only two parameters (forward and angular speeds). These two parameters can be obtained from wheel encoders. Since IMU sensors are typically expensive and the wheel encoders are only useful for the planar cases, using a single camera to estimate the relative motion of the camera comes into

consideration. If only a monocular camera is available, the absolute motion of the camera cannot generally be estimated. What could be estimated are the three angular speeds and the direction of the translation (not its amount). A well-known approach to calculate the motion of a camera up to scale is to obtain a 3×3 matrix (called essential matrix) from a set of correspondent points between two images.

The oldest and simplest method in this family is the 8-point method [LH81] in which the eight elements of the essential matrix are assumed to be independent and then by using eight correspondent points, a system of linear equations including eight equations is obtained. The number of the parameters of 6D motion up to scale is five. Therefore, five independent equations could be sufficient for their estimation. This fact shows that essential matrix elements are not independent. This issue can give rise to a poor performance of the 8-point method if there are measurement noises of magnitude more than half a pixel. In [Har97], a 7-point method is proposed, in which the dependencies of the essential matrix elements are taken into account to some degree. The 7-point method, however, does not perform better than the 8-point method significantly. In [Nis04], Nister managed to impose the dependency constraint of the essential matrix elements to come up with a 5-point method which requires only five points to recover camera motion up to scale. Nister showed that the 5-point method has a better performance in the presence of measurement noises. Nevertheless, this method is very time consuming as it necessitates symbolic processing techniques.

We investigate the reasons of the poor performances of the 8- and 7-point methods and propose some techniques to make the methods much more robust against measurement noise. To this end, the effect of noise on the relevant equations is considered.

1.3 Data Fusion

The purpose of data fusion is to use data of a feature tracking module to confine odometry errors. Using the pinhole model of a camera, it can be verified that the projection of a point in space on the retina of a camera depends on the relative position of the camera and the point. However, the relation is not straightforward

and the depth of the point cannot be calculated using a monocular camera. As a result, the only available measurements are bearing measurements.

From geometry it is known that the position of a point in space can be uniquely determined if at least two observations of the point at two different positions are available. Nevertheless, as discussed earlier, in reality, we cannot initially estimate the pose of a robot precisely due to the odometry noise. Therefore, the complexity of the monocular SLAM problem becomes more clear, in which unlike the usual SLAM algorithms, a depth estimation part also needs to be augmented in formulations.

Concerning SLAM algorithms, it should be mentioned that they leverage multiple observations of landmarks to obtain enough equations to determine parameters of robot poses and landmark positions and also minimize the effects of odometry and measurement noise. To this end, two approaches can be considered: the filter based approach and the cost optimization approach.

In the filtering approach, the SLAM problem is defined in the context of the probability theory and different realizations of Bayesian filters are used to estimate robot and landmark positions recursively in prediction and update phases. Essentially, two main filtering methods have been used to cope with the SLAM problem: Kalman filter [Kal60] and particle filter [GSS93] based methods. Kalman filters are applicable if the motion and measurement models are linear and also the processed random vector can be assumed Gaussian. With these assumptions, a Kalman filter estimates the mean and covariance of the vector using a recursive algorithm which includes prediction and update phases. The Kalman filter is known as an optimal estimation method for linear systems. Using Kalman filters for nonlinear systems can be feasible based on two methods. First, extended Kalman filters (EKF) which use linearized versions of the system equations about the operating point of the system. Second, unscented Kalman filters (UKF [JU04]) which use some points on the boundaries of Gaussian distributions to propagate in nonlinear equations. The EKF has been used for the mobile robot localization problem firstly in [SSC90] and its convergence was discussed and proved in [DNC$^+$01]. Since the complexity of the algorithm grows quadratically with respect to the number of landmarks (due to the size of the

covariance matrix used in the Kalman filter), some later works developed different formulations based on the assumption that the landmark positions are independent [GN01], [LJF00] and [WDD02]. Using linear approximations of motion and measurement models in an EKF typically gives promising results as long as the robot does not rotate noticeably. If the robot rotates or the landmarks are very close to the robot, the linear approximation can cause the divergence of the algorithm. Therefore, numerical approaches came into attention, by which the nonlinear differential equations are handled. The first attempt to use a numerical approach to the SLAM problem is reported in [TBF+98]. They discretized the map of the environment on a grid and then defined probability density functions on the grid map. Since the estimation of the robot position depends on the map and the map estimation also depends on the robot position, they used an expectation maximization (EM) algorithm to estimate these two parts iteratively. The algorithm, however, was developed for offline cases. Therefore, Murphy in [Mur00] firstly extended the EM algorithm for real time scenarios by windowing data. He showed that using limited size of data, the EM algorithm often gets stuck in local minima. Later on, he used Rao-Blackwellised particle filters to address the SLAM problem in a grid map by ignoring the robot orientation.

In case of non-Gaussian distributions, particle filters [GSS93] are variants of Kalman filters. In particle filters, the estimated PDF (probability density function) of a vector is represented by a set of weighted samples (particles). The particles are propagated in a prediction phase through a motion model and their weights are modified by receiving new measurements in the update phase. The main problem concerning particle filters is the exponential increase of their complexity with respect to the dimension of the estimated vector. Since the positions of landmarks in a map are usually assumed constant, their evolution model is linear. Hence, this problem can be handled by Rao-Blackwellised particle filters [CR96]. A Rao-Blackwellised particle filter is a variation of particle filters in which the estimation of the linear part is handled by the Kalman filter and the PDF of nonlinear part is modeled by particle filters.

In [FBDT99], Fox et al. discussed the global localization of a robot using particle filters in a continuous pre-known map. Later, they extended their work to address the SLAM problem using Rao-Blackwellised particle filters. They decomposed

the joint probability of robot and landmark positions into two different parts and then by assuming that the landmark positions were independent from each other, for each pair of particles and landmarks, an EKF was used to handle uncertainty of a landmark. They called their approach FastSLAM [MTKW02]. In [MTKW03], the prediction part of FastSLAM was modified to include the last measurements to generate particles more efficiently.

The discussed filters, nevertheless, are not applicable for the monocular SLAM problem. Since in both EKF based and Rao-Blackwellised based methods, landmarks uncertainties are modeled using Gaussian distributions; while for the monocular SLAM problem, the Gaussian assumption is not valid anymore since the depths of the landmarks are not known initially. To address, this problem two main approaches have been proposed. First, the delayed method in which initially the depths of landmarks are estimated and then based on the approximated depths the landmark uncertainties are initialized by Gaussian distributions [Bai03]. Obviously, the delayed method can result in more accumulated error while the run of the SLAM algorithm is postponed. Hence, the un-delayed methods have been proposed, in which landmark positions are parametrized such that the parameters can be modeled using Gaussian distributions when they are observed for the first time. In [KHH$^+$05], landmark uncertainties are modeled by mixtures of Gaussian distributions. The authors used several parallel EKF filters initialized at different combinations of depth assumptions for the landmark positions. These assumptions, however, cause the complexity of the algorithm to increase exponentially depending on the number of landmarks or the maximum assumed depth for a landmark position. In another method proposed in [SMDL05], given the first observation of a landmark, separate landmarks at different depths are initialized in a Kalman filter. Consequently, the problem is formulated using just one EKF filter. This method suffers from inconsistencies in landmark associations in different views. Civera et al. have introduced another technique to deal with the monocular SLAM problem in the context of EKF filters [CDM08]. They showed that if they parametrize landmark positions based on inverse depth parameters rather than their absolute positions, it is possible to model the uncertainties of these parameters with Gaussian distributions. This method has a poor performance if the landmarks are initialized near to the robot

(less than three meters). The reason of this poor performance lies in the severe nonlinearities of the inverse depth parametrization model at the depth one. In this case, the inverse depth parametrization method maps the depth uncertainties between 1 to $+\infty$ to the interval [0,1] and on the other hand it maps the interval [0,1] to 1 and $+\infty$. It means that in the neighborhood of the depth one, a severe nonlinearity exists, which cannot be handled by linear approximations needed in EKFs. Additionally, if landmarks are observed in low parallax angles, it occurs very often that the landmarks get localized behind cameras (negative depth) resulting in the fast divergence of the algorithm. The work proposed in [PJ08] uses a logarithmic depth parametrization to avoid the negative depth problem. The authors of this work used a second order EKF since with the logarithmic depth parametrization, the distributions of landmark positions are not near to Gaussian distributions any more. Using this technique, however, does not solve the problem of high nonlinearities in the measurement model for close landmarks.

Unlike filtering based methods, cost optimization based methods do not rely on linearized models, but rather they minimize a cost function between the predicted and real measurements by updating camera poses and landmark positions iteratively. In this approach, however, parametrization of landmark positions is still an important issue such that both close and far landmarks can be leveraged. In environments where there are large depth variations (such as outdoor environments), using a normal Cartesian parametrization for landmark positions degrades the estimation of the robot and landmark positions significantly. As the cost functions have generally several local minima and the initial guesses determine the final solutions. In [SMD10], the inverse depth parametrization is used in the context of a cost optimization method. Hence, this method also suffers from the negative depth problem. In [ZHYD11], a delayed method based on the parallax angle parametrization is proposed to deal with the negative depth problem. This method can work with the landmarks observed at high parallax angles. Hence, it fails if it cannot observe enough landmarks at high parallax angles.

1.4 Contributions of this Work

We contribute to the monocular SLAM problem in two parts: visual odometry and data fusion. Concerning the visual odometry part, firstly, we evaluate different feature tracking techniques based on the large KITTI dataset for visual odometry. Then we discuss why the 8-point method has a poor performance in the presence of measurement noise and push forward some modifications to make the method work robustly against measurement noise of the magnitude more than two pixels. Additionally, a regularization constraint based on the physical properties of cameras will be obtained and augmented in different N-point methods to especially enhance the estimation of rotation matrices. We also propose a new method to estimate the absolute scale factors of camera motions if the cameras are installed on wheeled vehicles with known heights over the ground plane.

The second contribution to the monocular SLAM problem is development of new fusion methods based on particle and UKF filters. As discussed in section 1.3, the main challenge of monocular SLAM is how to model large depth uncertainties of landmark positions at the initialization time. Among the discussed methods, the method based on the Gaussian sum filter (GSF), inverse depth parametrization and logarithmic parametrization methods have solid theoretical backgrounds. Nevertheless, each of them has its own shortcomings either in the sense of complexity or in the sense of robustness. To address these problems, for the case of planar motions, we propose new methods based on particle filters in such a way that the uncertainty of a robot pose is handled by particles and attached to each particle for each landmark, a line segment is assumed, of which length models the landmark's uncertainty. As the robot moves and makes new observations of the landmark, the lengths of line segments are reduced and the robot pose uncertainty is minimized through the trimmed line segments and new observations. The high performances of the proposed methods will be proven through demanding simulated and real experiments.

1.5 Thesis Outline

The thesis is organized as follows: in chapter 2, a comprehensive survey of previous works concerning visual feature tracking, visual odometry and fusion methods is presented. In chapter 3, feature tracking methods are investigated and then new robust visual odometry methods will be proposed. The methods are evaluated through simulation and application to the demanding KITTI dataset for visual odometry [KIT15]. In chapter 4, we push forward new fusion techniques for the 2D monocular SLAM problem. The methods are evaluated in comparison to previous methods based on simulated and real experiments.

2 Related Literature

As mentioned in the introduction chapter, the monocular SLAM problem consists of three main parts: feature tracking, odometry and data fusion. In this chapter, we conduct a comprehensive survey of these topics.

2.1 Feature Extraction and Association

In the terminology of image processing, a feature in an image is a point or set of points which have special characteristic with respect to their neighborhoods and can be distinguished from them. Edges, corners and blobs can be counted as low level features. Feature extraction methods constitute an important and active field of research in the image processing although it has been investigate for more than two decades. One important application of feature extraction is tracking of image elements, which can be used for higher level analysis of a dynamic scene or for the estimation of camera motion.

The feature tracking problem can be approached by two different ways. First, image matching methods in which given two images, their features are extracted separately and then the correspondent features are found based on local descriptors of the features. Several works have been done in this regard, but here we discuss the two most cited methods which have been applied in various image processing applications in the last decade: the Scale invariant features transform (SIFT) and the speeded up robust features (SURF) methods. It is good to mention that, in the recent years, inspired from SIFT and SURF, new feature matching methods have been proposed, which mainly target the speed of feature matching while trying to keep the precision of matching near to the SIFT and SURF methods. The methods are based on binary patterns in neighborhoods of

features. The method can be counted as: BRIEF (binary robust independent elementary features) [CLSF10], ORB (oriented robust brief) [RRKB11] and BRISK (binary Robust invariant scalable keypoints) [LCS11].

The second approach for feature tracking is based on optical flow methods in which motion of a point in an image sequence is calculated with the assumption that the intensity of the point does not change within the sequence.

In this section, we firstly discuss the image matching methods and the influential theory behind them (scale space analysis), and then the optical flow based methods will be introduced.

2.1.1 Scale Space Filtering

In digital signal processing, the description of a signal (e.g. $f(x)$) in a compact way has been investigated for several years. This problem has been approached mostly by finding some primitive descriptors or features from a signal by which the original signal can be reconstructed.

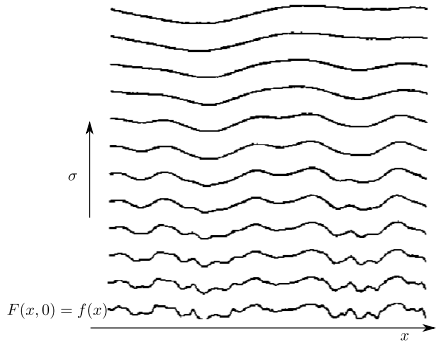


Figure 2.1: Convolution of a signal with Gaussian functions with different variances (from [Wit83]).

Obviously, an important group of features can be the local extrema of the signal and its derivatives (mostly first and second order derivatives). Consequently,

it should be determined with which resolution or scale, we are interested to reconstruct the signal. By scale, we mean how much details of the signal are important for us. Are we interested in fine changes of the signal (low scale) or coarse variations (high scale). To have a general answer to this question a transform was offered by Witkin in [Wit83], in which a signal is convolved by a Gaussian function of which the standard deviation is the scale parameter.

$$F(x,\sigma) = f(x) * g(x,\sigma) = \int_{-\infty}^{+\infty} f(u) \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-u)^2}{2\sigma^2}} du \tag{2.1}$$

As shown in Fig. 2.1, by increasing the variance of the Gaussian, more details of the signal disappear. The feature points can be obtained at the scale value σ by assigning the n^{th} derivative of the smoothed signal to zero and solving the equation:

$$\frac{\partial^n F}{\partial x^n} = f * \frac{\partial^n g}{\partial x^n} = 0 \tag{2.2}$$

Witkin used special extrema achieved from the second derivative which are known as inflection points:

$$F_{xx} = \frac{\partial^2 F}{\partial x^2} = 0; F_{xxx} = \frac{\partial^3 F}{\partial x^3} \neq 0 \tag{2.3}$$

Fig. 2.2 shows the contours of inflection points. By increasing scale, we can see how the inflection points approach each other, merge together and finally disappear. Additionally, it can be seen that the feature points have displacements with respect to their original position in the finest scale. Hence, if we intend to represent a signal in coarse scale, the displacement of inflection points could be an issue. As a result, a method is required to localize the features in the original signal. Obviously, based on the contours diagram an inflection point can be tracked to its position in the original signal. Witkin also proposed a chart called tree diagram to show how a feature survives or disappears while scale increases (Fig. 2.3).

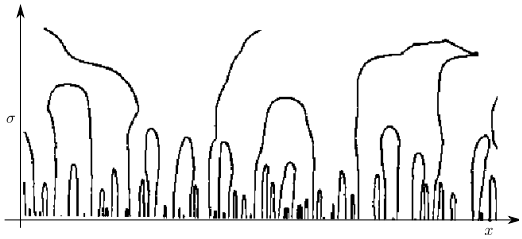


Figure 2.2: The contours of inflection points (from [Wit83]).

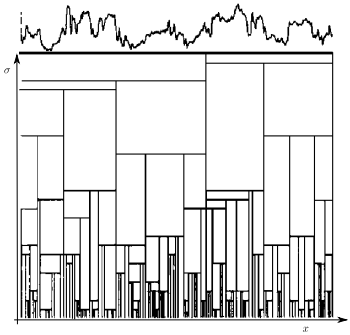


Figure 2.3: Tree diagram to show the appearance of the inflection points at different scales (from [Wit83]).

The scale space analysis was extended for images (2D signals) by Koendernik in [Koe84] to extract the structures of an image at different scales. He tried to find a kernel function and proper features by which the following goals can be achieved:

1. Deriving a one parameter (scale) family from an image.

2. Studying the structure of each family member by considering their relations.

His formulation was for gray scale images, and as we know, a gray scale image can be defined by a function such as $L : \mathcal{R}^2 \rightarrow \mathcal{R}$ where:

$$L(r) = L(x, y) = \lambda; \quad r \in \mathcal{R}^2, \lambda \in \mathcal{R} \tag{2.4}$$

Then, for such a function a scale-space function can be defined:

$$L(x, y; s) = \Lambda; \quad \Lambda \in \mathcal{R} \tag{2.5}$$

where $s = \sigma^2$ is the scale parameter ($L(x, y; 0) = L(x, y)$). Then, he tried to find a sensible way to generate the one parameter family by assuming a condition called causality; meaning, the features at a coarse resolution should have causes (not necessarily unique) in finer resolution. In other words, if we increase scale, we expect the features to disappear or not change. It means that new features should not emerge. Based on this assumption, he inferred that the scale space function should satisfy the heat or diffusion equation:

$$\nabla^2 L = L_{xx} + L_{yy} = L_s \tag{2.6}$$

where $L_{xx} = \frac{\partial^2 L}{\partial x^2}$, $L_{yy} = \frac{\partial^2 L}{\partial y^2}$, and $L_s = \frac{\partial L}{\partial s}$. A solution to this equation is the convolution of the image with Gaussian kernel with the variance s:

$$L(x, y; s) = L(x, y) * g(x, y; s) \tag{2.7}$$

where $g(.)$ is:

$$g(x, y; s) = \frac{1}{\sqrt{2\pi s}} e^{-(x^2+y^2)/2s} \tag{2.8}$$

The next step is to find some borders to classify an image at different scales. For one dimensional signals, inflection points were selected. The 2D equivalent of inflection points are parabolic curves:

$$L_{xx} L_{yy} - L_{xy}^2 = 0 \tag{2.9}$$

But he discussed that the parabolic curves cannot always enclose single extrema. He also claimed that zero crossing curves obtained from the Laplace equation:

$$\nabla^2 L = L_{xx} + L_{yy} = 0 \tag{2.10}$$

suffer from the same problem. To address this problem, he used saddle points in blurred images and curves created from the intersection of the blurred images and a plane which passes through the saddle points. The saddle points can be obtained from a Hessian Matrix which is defined at each point of the image L as follows:

$$H(L) = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{bmatrix} \tag{2.11}$$

The saddle points are the points in which the Hessian matrix becomes indefinite.

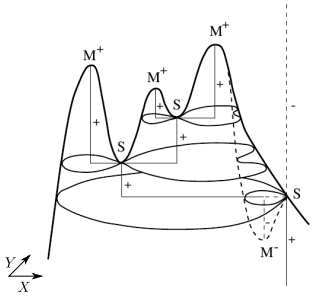


Figure 2.4: Saddle points in scale-space (from [Koe84]).

Using a Gaussian kernel to create scale-space family has the following important property:

$$L(.; s_1) * g(.; s_2) = L(.; s_1 + s_2) \tag{2.12}$$

The scale space theory has two main applications. First, image segmentation as discussed before and second finding features in an image which are invariant with respect to scales. In other words, features which survive in a long range of scale variations.

As already discussed, the best features in an image are saddle points, however, finding saddle points needs the calculation of the Hessian matrix for every point which would be very time consuming. To solve this problem it can be verified that the saddle points should be located mostly on the edges of an image. Therefore

we can firstly use an edge detection method such as Laplace of Gaussian (LoG) to find the edges in an image and then determine the saddle points among the edges.

$$\{(x, y)|\nabla^2 L(x, y; s) > \delta\} \tag{2.13}$$

On the other hand, if we consider the diffusion equation (Eq. (2.6)), it can be inferred that the LoG can be obtained from the differentiation of $L(.; s)$ with respect to s.

2.1.2 Pyramid Analysis

The idea of finding features at different scales has been approached by a simpler way which is sampling and smoothing the image recursively. It is known that sampling may result in artifacts due to aliasing which may appear in the sampled version of a signal. To alleviate this problem, images should be smoothed with smoothing kernels before the sampling process is performed.

Pyramid analysis was firstly introduced by Burt in [Bur81] for one dimensional discrete signals. Given the signal $f(x)$, he proposed the following recursive formula to establish the pyramid:

$$g_l(x) = \sum_{i=-m}^{m} w(i) g_{l-1}(x + ir^{l-1}); \quad l > 1 \tag{2.14}$$

where $g_0(x) = f(x)$, r is the sampling interval, l is the level of the pyramid, $2m + 1$ is the width of kernel and $w(.)$ defines the shape of the kernel.

From Eq. (2.14), it can be obtained:

$$g_l(x) = h_l(x) * f(x) = \sum_{i=-M_l}^{M_l} h_l(i) f(x + i) \tag{2.15}$$

where $h_l(x)$ is called equivalent kernel at level l and $2M_l + 1$ is the equivalent kernel width which should be calculated.

$h_l(x)$ and M_l can be calculated as follows:

$$h_l(x) = \sum_{i=-M_l}^{M_l} w(i) h_{l-1}(x - ir^{l-1}) \tag{2.16}$$

where $h_0(x) = \delta(x)$. The width of the equivalent kernel is obtained as follows:

$$\begin{aligned} M_l = mr^{l-1} + mr^{l-2} + ... + m &= m\sum_{i=0}^{l-1} r^i \\ &= m\frac{r^l - 1}{r - 1} \end{aligned} \tag{2.17}$$

Then, he assumed four constraints on the kernel such that the equivalent kernels remain unimodal, symmetric and centered at $x = 0$:

$$\sum_{i=-m}^{m} w(i) = 1 \tag{2.18}$$

$$w(x) = w(-x) \tag{2.19}$$

$$w(x_1) \geq w(x_2) \quad \text{for} \quad 0 \leq x_1 \leq x_2 \tag{2.20}$$

$$\sum_{i=-m}^{m} w(j + ir) = 1/r \quad \text{for } j: \quad 0 \leq j < r \tag{2.21}$$

A special case which is used mostly in pyramid analysis is the kernel size 5 ($m = 2$). Imposing the conditions for this case and by assuming $w(0) = a$, $w(1) = b$, $w(2) = c$, we have:

$$\begin{aligned} &w(-1) = w(1) = b \\ &w(-2) = w(2) = c \\ &a + 2b + 2c = 1 \\ &a \geq b \geq c \geq 0 \end{aligned} \tag{2.22}$$

which results in:

$$\begin{aligned} 1/4 \leq a \leq 1/2 \\ b = 1/4 \\ c = 1/4 - a/2 \end{aligned} \tag{2.23}$$

For different a different kernels can be created (Fig. 2.5). For $a = 0.4$, the equivalent kernel approaches to a Gaussian function as $l \to \infty$ and for $a = 0.5$, it approaches to a triangle.

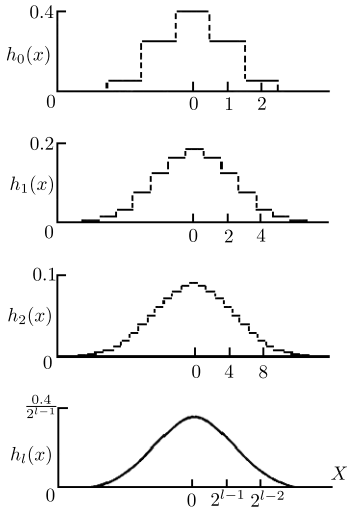


Figure 2.5: Equivalant kernels at different levels (from [Bur81]).

Burt extended his approach for two dimensional signals (images) in [BA83]. Assuming the source image $L(x, y)$ and the kernel $w(m, n)$ with the size 5×5 the recursive images can be obtained as follows:

$$G_l(x, y) = \sum_{i=-2}^{2} \sum_{j=-2}^{2} w(i, j) G_{l-1}(2x + i, 2y + j) \tag{2.24}$$

where $G_0(x, y) = L(x, y)$.

To find the weights of the kernel, he assumed the kernel to be symmetric again. Therefor, $w(m, n)$ can be decomposed as follows:

$$w(i, j) = w_1(i)w_2(j) \tag{2.25}$$

Consequently, each of $w_1(i)$ and $w_2(j)$ can be determined using the one dimensional case. For $w_1(0) = w_2(0) = 0.4$ a 2D Gaussian is obtained. In Fig. 2.6 the result of applying Gaussian pyramid on the Lena image for five levels can be seen.

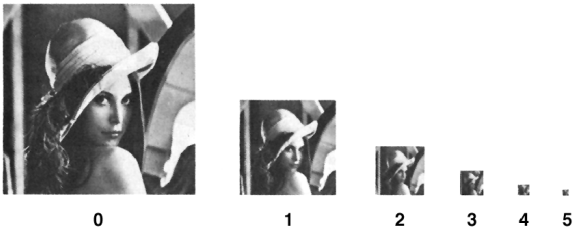


Figure 2.6: Calculating Gaussian pyramid for Lena image (from [BA83]).

2.1.3 SIFT

Looking for features which are stable at different scales or in other words scale invariant, Lowe used the idea of scale space theory and pyramid analysis to create a variety of blurred and sub sampled images to find stable extrema points among them [Low04].

He created a pyramid with four levels (octaves) and then blurred the images at each level several times to cover the large gaps between the pyramids levels. He used the sampling rate of two for the pyramid and blurred the base image recursively in each level (octave) five times as follows:

$$L_s(x, y) = G(x, y; k^{s-1}\sigma) * L_{s-1}(x, y); \quad s = 1...5 \tag{2.26}$$

where $L_0(x, y)$ is the base image in each octave and k is chosen $\sqrt{2}$ in order to cover the gaps between the octaves with five blurred images. $G(x, y; \sigma)$ is a 2D Gaussian function with the standard deviation of σ. Reminding the diffusion equation Eq. (2.6), he used a DoG between the images in an octave to obtain stable extrema:

$$\sigma \nabla G = \frac{\partial G}{\partial \sigma} = \frac{G(x, y; k\sigma) - G(x, y; \sigma)}{k\sigma - \sigma} \tag{2.27}$$

and then:

$$D = G(x, y; k\sigma) - G(x, y; \sigma) = (k - 1)\sigma^2 \nabla^2 G \tag{2.28}$$

Fig. 2.7 shows how the pyramid, the blurred images and their differentiations are made:

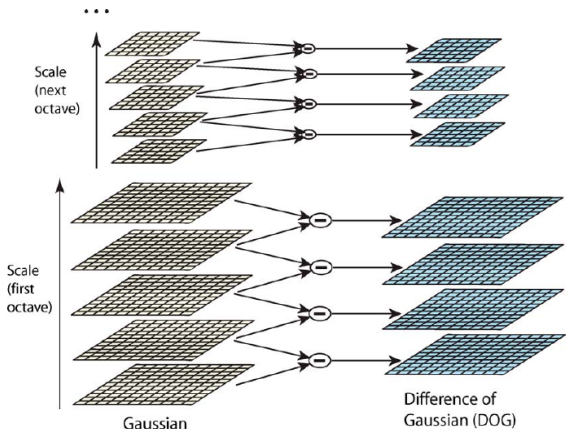


Figure 2.7: Establishing pyramid and scale space and creating DoG images (from [Low04]). The left side shows blurred images at different octaves and the right side depicts the differences of blurred images.

After obtaining DoG images, the extrema (keypoints) are extracted in such away that at any scale the intensity of a point in a DoG image should be greater than

the intensities of the eight surrounding points and also greater than the intensities of the eighteen points in upper and lower scales (Fig. 2.8).

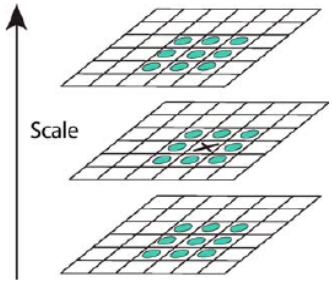


Figure 2.8: Extrema points (from [Low04]).

Lowe conducted several tests to find out a proper value for σ by which the base image get blurred at each octave such that the features appear under different transformations and noise levels more (repeatability). And he found out that for $\sigma = 1.6$ the best repeatability occurred.

To have keypoints which are more robust against noise, Lowe used a threshold value to reject the points at which the image had low contrasts. Assuming the intensity of the image varies between [0,1], he experimentally found the proper threshold value to be 0.03.

The DoG operator creates also strong responses at the points located on the edges. These points, however, cannot be determined uniquely since they are not distinctive from the other points located on the same edge in their neighborhoods. This problem is known as aperture problem in image processing literature. To reject these points, Lowe used the following Hessian matrix:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix} \tag{2.29}$$

and considered the fact that the ratio of eigenvalues of the matrix are proportional with curvature at the point and since the points on the edges cause strong

curvatures just in one direction, one of the eigenvalues will be much greater than the other. If the larger eigenvalue is α and the smaller β, the ratio $r = \frac{\alpha}{\beta} > 10$ could be a simple measure to eliminate the extrema on the edges.

SIFT Correspondences

Image matching means that from two different images several keypoints are extracted and then the correspondences between the points from two images are found. Lowe extracted a special histogram based on the neighborhood of each keypoint and by comparing the histograms the correspondent points are found.

For each keypoint which is extracted at a special scale, the nearest blurred image is selected to define the associated histogram for the keypoint. In the selected blurred image, at each point, the following values named as magnitude and orientation should be calculated:

$$\begin{aligned} m(x,y) &= \sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2} \\ \theta(x,y) &= \tan^{-1}\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)} \end{aligned} \tag{2.30}$$

As can be observed in Fig. 2.9, at each point, a vector based on the calculated magnitude and the orientation can be defined. The vectors located in a squared window (8×8) centered at each keypoint are used to form the histograms. In this regard, the window is divided into four sub windows. Then, in each sub-window eight bins are considered to create a histogram. Meaning, eight intervals for orientation values are chosen([0° 45°], [45° 90°], [90° 135°], [135° 180°], [0° -45°], [-45° -90°],[-90° -135°],[-135° -180°]) and the weights of each vector is added to the related bin. This version of SIFT is called SIFT32. Lowe showed that if the size of the main window is increased to 16×16, the outlier ratio decreases noticeably. In this case sixteen 4×4 sub-windows exist, which result in a descriptor vector of the length 128.

By associating a descriptor vector to each keypoint, the matched points can be found by comparing the histograms (descriptors) of each two point from two

images. The features which have minimum euclidean distances and are also less than a threshold (0.3) can be considered as the matches.

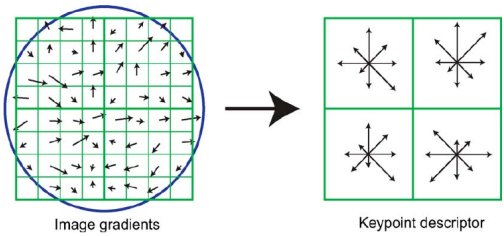


Figure 2.9: Magnitude and orientation of the points around a keypoint (from [Low04]).

To make SIFT invariant to rotations, the orientation of brightness gradient at the keypoint is used as a reference orientation and the histogram is rearranged based on this orientation.

The SIFT method has a robust performance against illumination change and noises; however, it is computationally expensive in all the three main parts of the method: blob extractions, descriptor computations and feature matching. Image smoothing using Gaussian kernels for the purpose of blob extraction imposes relatively high computation loads. On the other hand, computation of histogram of gradients is time consuming due to the need for the calculations of gradient angles. Finally to find the best match for a feature in the first image all features in the second image should be compared with the feature in the first image, which is also a very time consuming process. Another issue concerning the SIFT method is the repeatability. This problem stems from the discrete sampling of scale-space and it is possible that a feature which has been detected in the first image may not be detected in the second image since it is not located at the sampled scales. The three first problems, as we will see, will be addressed by the SURF method to some extent. Nevertheless, the repeatability problem is a common problem among all feature matching methods. An open source implementation of SIFT for C++ is available in OpenCV library [Ope15], which we used to evaluate SIFT for feature tracking in chapter 3.

2.1.4 SURF

Inspired by the SIFT algorithm, Herbert et al. offered a faster method for the detection of scale invariant features [HTG08]. They made some variations in different parts of the SIFT algorithm, of which the important one was using the approximations of second order Gaussian derivatives as the smoothing kernels. The approximations are similar to the Haar wavelets. As depicted in Fig. 2.10, the kernels include three regions: positive, negative and zero, the positive and negative regions are called lobs with the length l_0. The kernels shown in Fig. 2.10 are the basic kernels from which kernels for higher scales ban be generated. To create a kernel at a higher scale from the lower scale, one point should be added to the each side of each lob in the lower scale kernel. Using such kernels gives the possibility to smooth the source image several times faster by applying an integral image which is defined as follows:

$$I(x,y) = \sum_{i=0}^{x} \sum_{j=0}^{y} L(i,j) \tag{2.31}$$

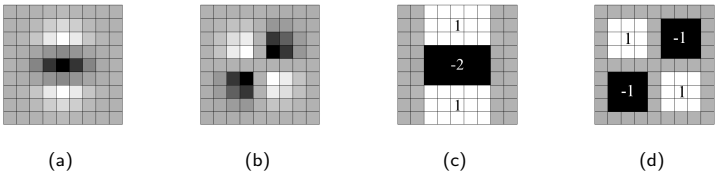


Figure 2.10: Second order Gaussian derivatives: (a) L_{yy}, (b) L_{xy}, (c) and (d) their wavelet approximations (from [HTG08]).

Once an integral image is calculated, the convolution of an image with a wavelet kernel can be obtained by only four summations. Using Haar wavelets in the directions x and y, the values L_{xx}, L_{yy} and L_{xy} at any point are obtained, by which a Hessian matrix can be formed as follows:

$$H = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{bmatrix} \tag{2.32}$$

To determine an extrema point in scale-space, the approximated determinant of H is considered:

$$|H|_{approx} = L_{xx}L_{yy} - (wL_{xy})^2 \tag{2.33}$$

where $w = 0.91$ is selected to equalize the kernel energy with the Gaussian kernel. In the first octave, kernels with the sizes of 9, 15, 21 and 27 are used to create the scales. If Eq. (2.33) results in a maximum or minimum value at a point within a cube of the size $3 \times 3 \times 3$ in both scale and space, the point is selected as an extremum.

To build up the octaves, SURF only increases the size of the kernels; unlike SIFT which sub-samples the image. Hence, it alleviates the aliasing problem caused by image sub-sampling. To create proper kernels for higher scale, the sizes of the lobs of the kernels are doubled. Accordingly, in the second octave the kernel sizes are 15, 27, 39 and 51 and in the third octave are 27, 51, 77 and 99. Searching the extrema points in upper octaves is done by sampling the smoothed images by the factor 2^i where i is the octave number.

After detecting extrema points in scale and octaves, descriptor vectors should be assigned to them. To make SURF invariant to rotations, firstly, similar to SIFT, the reference (dominant) orientation for each keypoint is obtained. In this regard, SURF uses first order Haar wavelets shown in Fig. 2.11 to calculate the orientation vectors $[\frac{dL}{dx} \ \frac{dL}{dy}]^T$ at a circular vicinity of each keypoint. The size of the vicinity is chosen to be $6s$ where s is the scale factor. Obviously, it is not necessary to calculate the gradient values at every point in all scales and the points can be sampled by the factor of s. After the calculation of the gradients, they should be weighted by a Gaussian kernel with the standard deviation $\sigma = 2s$, which is centered on a keypoint.

After calculation of the vectors at each point, the dominant direction is found by summing up the vectors in a window which is a $\frac{\pi}{3}$ sector of the circular vicinity of the keypoint. Consequently, the summed vector which has the greatest length

is selected as the dominant vector which determines the dominant direction (Fig. 2.12).

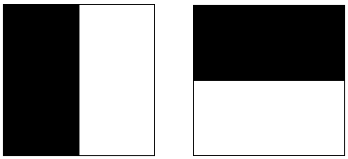


Figure 2.11: First order wavelets in the direction x and y (from [HTG08]).

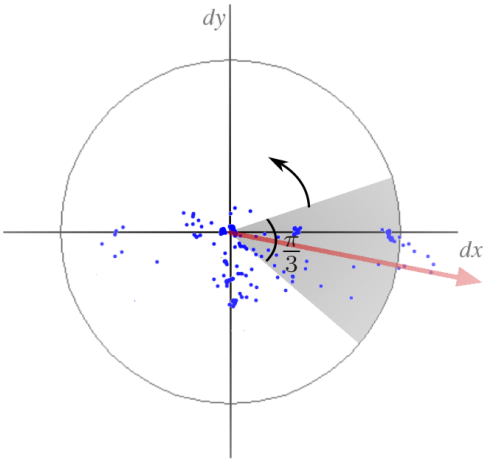


Figure 2.12: How to find the dominant change of the illumination at a keypoint (from [HTG08]).

SURF descriptors are obtained by assuming a square window centered on each keypoint directed with the dominant orientation which is already obtained. The size of the window is $16s \times 16s$, which is divided into $4s \times 4s$ sub-windows (Fig. 2.13). By applying first order Haar wavelets on each sub-window, the following values are obtained:

$$\sum dx, \quad \sum |dx|, \quad \sum dy, \quad \sum |dy| \tag{2.34}$$

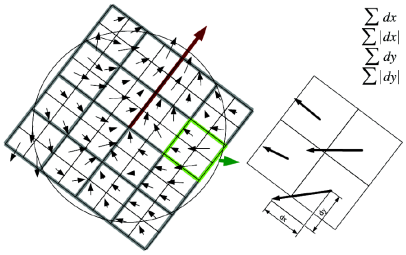


Figure 2.13: The extraction of Surf descriptors (from [HTG08]).

By accumulating these values in a vector, a descriptor vector with the size of 64 is generated.

SURF improved the speed of blob detections and also descriptor calculation to a great extent thanks to applying integral images and Haar wavelets. However, since Haar wavelets are approximations of DoG operators, the calculated descriptors are not as good as SIFT descriptors resulting in higher rate of outliers. Additionally, the blob centers cannot be well localized. Blob localization is based on the gradient of an image. Since the weights of the Haar kernels in their central regions are the same, image gradients cannot be calculated well.

An open source implementation of SURF for C++ is available in OpenCV, which was used for its evaluation in chapter 3.

2.1.5 CenSurE

Agraval et al. discussed in [AKB08] that the SIFT and SURF points are not accurate enough for the purpose of feature tracking. In fact, the points are the centers of blobs which cannot be localized in the original image exactly if the blobs are detected in high scales. On the other hand, as mentioned before, corners are not stable features against the scale changes and additionally they are not

also completely rotational invariant. Therefore, they used again the idea of SIFT but instead of sub-sampling images, they offered a special kernel called Center Surrounding Extrema (CenSurE) which simulates the Laplace of Gaussian but with a constant cost of calculations at different scales. The proposed kernels are boxes, hexagons and octagons which all have inner and outer regions (Fig. 2.14). The sum of the weights of the each kernel should be zero.

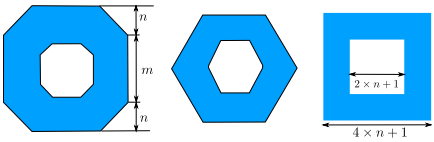


Figure 2.14: Censure kernels (from [AKB08]).

They clarified that box kernels are not rotation invariant and therefore offered two other kernels. Nevertheless, using the other kernels, they could not use the advantage of integral images which make the cost of a convolution independent of the size of the kernel. To address this problem, they came up with a modified version of an integral image called slanted integral image as follows:

$$I_\alpha(x,y) = \sum_{j=0}^{y} \sum_{i=0}^{x+\alpha(y-j)} L(i,j) \tag{2.35}$$

In Fig. 2.15, we see how a trapezoidal region is extracted using the slanted integral image.

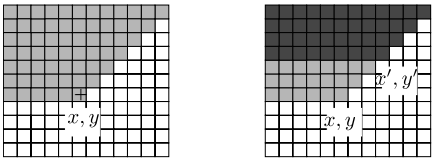


Figure 2.15: Left: Slanted integral image. Right: Calculation of the summation of the weights in a trapezoid located between two corners (x, y) and (x', y') (from [AKB08]).

It can be verified that the summation over a trapezoid can be done by four summations. On the other hand, it can be shown that a box would be decomposed into one trapezoid for $\alpha = 0$, a hexagonal into two trapezoids with $\alpha = \pm\sqrt{3}$ and an octagonal into three trapezoids with $\alpha = \pm 1, 0$.

The CenSurE feature detectors are implemented in OpenCV; however, to best of out knowledge no free implementation for the calculation of their descriptors is available.

2.1.6 Binary Descriptors

In recent years, instead of gradient based methods introduced before, new methods have been proposed which generate binary descriptors based on the local brightnesses about each keypoint. These methods mainly target the speed of the descriptor generation and also matching processes. Concerning the matching process, these methods use XOR operation and bit counting to obtain distances between descriptors.

The first method in this family is BRIEF (binary robust independent elementary features) [CLSF10]. In this method, based on different pairs of test points in a rectangular neighborhood of a keypoint, a descriptor with the length 32 or 64 is created. The authors proposed five different patterns to sample test points from a neighborhood (Fig. 2.16).

Given an image L and each pair of points in the image such as $\mathbf{x} = (x_i, y_i)$ and $\mathbf{y} = (x_j, y_j)$ of test points a bit is generated as follows:

$$\tau(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & if \quad L(\mathbf{x}) > L(\mathbf{y}) \\ 0 & else \end{cases} \tag{2.36}$$

By stacking all generated bits based on all test points in a vector, a descriptor vector for a keypoint is generated.

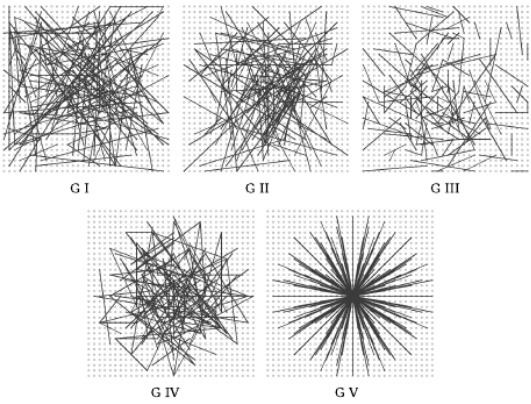


Figure 2.16: Five different patterns to sample test points in a neighborhood of a keypoint in the BRIEF method (from [CLSF10]).

The BRIEF is not rotation invariant. Therefore, in [RRKB11], a rotation invariant version of BRIEF known as ORB was proposed. In this method, based on the moments of intensities in a neighborhood of a keypoint, a rotation angle is assigned to each keypoint. Based on the obtained rotation, the test points in the BRIEF method are rotated about the keypoint.

In another work proposed in [LCS11], a circularly symmetric pattern for the selection of test points was utilized (Fig. 2.17). The method is known as BRISK (binary robust invariant scalable keypoins). In this method, the intensities of the test points are obtained based on averaging intensities in a circular neighborhood of each sample point. As can be seen in Fig. 2.17, the vicinity for the smoothing is increased proportional to the distances of the points from the keypoint. BRISK is computationally more expensive than ORB and BRIEF and concerning the quality of matching, it cannot provide better results.

As we will show in section 3, the binary descriptors are not as discriminant as SIFT and SURF. As a result, to have a good rate of correct matches, lots of matches should be dispelled.

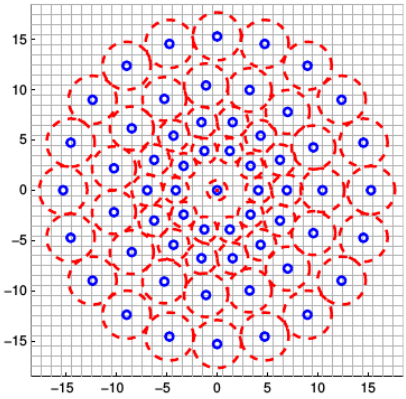


Figure 2.17: The test points used in the BRISK method and the vicinity to apply smoothing for each test point (from [LCS11]).

2.1.7 Optical Flow

Optical flow methods attempt to find the displacement vectors of all points between two images. The basic assumption in this regard is that the intensities of correspondent points do not change between two images. Hence, given a time varying image $L(x, y, t)$, we have:

$$L(x, y, t) = L(x + \Delta x, y + \Delta y, t + \Delta t) \tag{2.37}$$

On the other hand, if the point displacements are assumed small, a linear approximation for Eq. 2.37 based on the Taylor expansion can be applied:

$$L(x + \Delta x, y + \Delta y, t + \Delta t) = L(x, y, t) + \frac{\partial L}{\partial x}\Delta x + \frac{\partial L}{\partial y}\Delta y + \frac{\partial L}{\partial t}\Delta t \tag{2.38}$$

Using Eq. (2.37) and Eq. (2.38), the following equation is obtained:

$$\frac{\partial L}{\partial x}\Delta x + \frac{\partial L}{\partial y}\Delta y + \frac{\partial L}{\partial t}\Delta t = 0 \tag{2.39}$$

Consequently, by dividing both sides by Δt, we have:

$$\frac{\partial L}{\partial x}v_x + \frac{\partial L}{\partial y}v_y + \frac{\partial L}{\partial t} = 0 \tag{2.40}$$

where v_x and v_y are the displacements of a point in the directions x and y, which are unknown parameters required to be calculated. To determine the unknowns at least one more equation is required. Lucas and Kanade in [LK81] assumed that points in a small square vicinity with the size $(2w+1) \times (2w+1)$ have almost the same displacement vectors, meaning:

$$L_x(x+i, y+j)v_x + L_y(x+i, y+j)v_y + L_t(x+i, y+j) = 0 \tag{2.41}$$

where $L_x = \frac{\partial L}{\partial x}$, $L_y = \frac{\partial L}{\partial y}$ and $i, j = -w...w$.

Eq. (2.41) provides $(2w+1) \times (2w+1)$ equations which are more than two necessary equations. Therefore, we should approach the problem using the least squares method.

Eq. (2.41) can be presented in a matrix form as follows:

$$\mathbf{L_x v} = -\mathbf{L}_t(x, y) \tag{2.42}$$

where:

$$\mathbf{L_x} = \begin{bmatrix} L_x(\mathbf{p}_1) & L_y(\mathbf{p}_1) \\ \vdots & \vdots \\ L_x(\mathbf{p}_n) & L_y(\mathbf{p}_n) \end{bmatrix}; \quad \mathbf{L}_t = \begin{bmatrix} L_t(\mathbf{p}_1) \\ \vdots \\ L_t(\mathbf{p}_n) \end{bmatrix} \tag{2.43}$$

where $\mathbf{p}_r \in \{(x+i, y+j) | \, i, j = -w, ..., w\}$, $n = (2w+1) \times (2w+1)$ and $\mathbf{v} = [v_x \; v_y]^T$. Eq. (2.42) is least squares form equation and can be solved as follows:

$$\mathbf{v} = -(\mathbf{L_x}^T \mathbf{L_x})^{-1} \mathbf{L_x}^T \mathbf{L}_t \tag{2.44}$$

The constraint offered by Lucas and Kanade is a local constraint which has a poor performance for the points located at the edges. Additionally, if two occluding objects move differently the Lucas-Kanade method assumption would be violated at the occluding points.

Horn-Schunk, on the other hand, proposed a global constraint in [HS81] by defining a cost function as follows:

$$E(v_x, v_y) = \int \left[(L_x v_x + L_y v_y + L_t)^2 + \alpha(|\nabla v_x|^2 + |\nabla v_y|^2)\right] dx\ dy \tag{2.45}$$

where ∇ is the nabla operator and α is a smoothness factor.

By defining the energy function, the displacement vectors should be determined such that the function gets minimized. The minimization can be done using an iterative method:

$$\begin{aligned} v_x^{n+1} &= \bar{v}_x^n - L_x(L_x\bar{v}_x^n + L_y\bar{v}_y^n + L_t)/(\alpha^2 + L_x^2 + L_y^2) \\ v_y^{n+1} &= \bar{v}_y^n - L_y(L_x\bar{v}_x^n + L_y\bar{v}_y^n + L_t)/(\alpha^2 + L_x^2 + L_y^2) \end{aligned} \tag{2.46}$$

where $\bar{v}_x$ and $\bar{v}_y$ are average displacements which are calculated as follows:

$$\begin{aligned} \bar{v}_x &= \frac{1}{6}[v_x(x-1,y,t) + v_x(x+1,y,t) + v_x(x,y-1,t) + v_x(x,y+1,t)] \\ &+ \frac{1}{12}[v_x(x-1,y-1,t) + v_x(x+1,y+1,t) \\ &+ v_x(x-1,y+1,t) + v_x(x+1,y-1,t)] \end{aligned} \tag{2.47}$$

$$\begin{aligned} \bar{v}_y &= \frac{1}{6}[v_y(x-1,y,t) + v_x(x+1,y,t) + v_y(x,y-1,t) + v_y(x,y+1,t)] \\ &+ \frac{1}{12}[v_y(x-1,y-1,t) + v_y(x+1,y+1,t) \\ &+ v_y(x-1,y+1,t) + v_y(x+1,y-1,t)] \end{aligned} \tag{2.48}$$

Horn-Shunk discussed that α plays only role when the the gradient of image intensity in a part of the image is low. It means that at these points the displacement vectors are obtained from the average of the vectors at that neighborhood.

For tracking purposes, however, we do not require to obtain the displacement vectors at all points, but rather we look for stable features to track within the frames. Therefore, there has been lots of investigation in many works to find such features which are also invariant to the scale and rotation. As discussed before SIFT and SURF produce a class of such features which are the center of blobs at different scale and octaves. The problem concerning such features is imprecision of the features. Since by changing the sizes of the blobs in different frames, their centers may not be localized accurately especially if the features are extracted in high scale levels. Additionally, blobs are not proper features to be tracked by optical flow methods since the gradient of the illumination at their centers is almost zero. It can result in the singularity of the sparse optical flow method. On the other hand corner features, are appropriate features for tracking using the sparse optical flow method. Here we present two methods for the detection of corner features.

Harris corners

For the given image $L(x, y)$, a corner point is a point at which the gradient of image intensity changes at least in two directions. Harris and Stephens proposed a measure to determine if such changes occur at a point [HS88]. In this regard, a square neighborhood with the size of $(2n + 1) \times (2n + 1)$ is considered and the following matrix from intensities in the neighborhood is generated:

$$A(x, y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} w(i, j) \begin{bmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{bmatrix}_{x+i, y+j} \tag{2.49}$$

where $w(i, j)$ represents a weighting kernel (such as a Gaussian) with the size of $(2n + 1) \times (2n + 1)$. L_x and L_y also show the derivatives of L in the directions x and y. By calculating the eigenvalues of A, the point is considered as a corner if the both eigenvalues of A are nonzero and do not differ greatly. Harris corner detectors are quiet computationally expensive due to the need for calculation of eigen values. The repeatability of the detector is also not as good as SIFT or SURF feature detectors since the scale space analysis is not considered in the

Harris corner algorithm. Nevertheless, Harris corners are good features to be passed to the Lucas-Kanade method since the variations of image gradient are mostly enough to avoid singularities in Eq. (2.42).

FAST features

Harris corners are computationally expensive and therefore Rosten and Drummond proposed a new corner detector called FAST (Features from Accelerated Segment Test) which uses a simple criterion to find out if a point is a corner [RD05]. In FAST, a circle with the radius of three pixels is centered at each point and the intensities of the sixteen points located on the circumference of the circle are compared with the center point; subsequently, if at least twelve points on the circumference have intensities more or less than the center point, it is considered as a corner (Fig. 2.18).

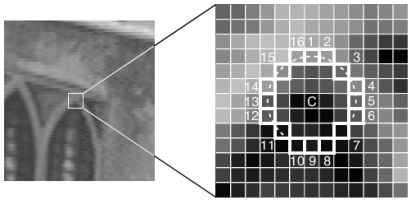


Figure 2.18: FAST corner detector.

FAST corner detectors are much faster than the Harris method; nevertheless, FAST may detect low quality features which are not appropriate to be tracked by the Lucas-Kanade method. Therefore, these features are mostly used in feature matching techniques [RRKB11].

2.2 Visual Odometry Using a Single Camera

Odometry in robotic literature refers to systems which provide instantaneous motion parameters of a rigid body. The free motion of a rigid body can be formulated by frame transformation methods to obtain its kinematic motion model in which only instantaneous speed parameters are used. These models are abstract models of which relations with the physical quantities such as forces or torques are not considered. A well-known odometry system for non-holonomic robots which have planar motions is using wheel encoders to convert the angular wheel velocities into a heading and a rotational velocities of the robot. Clearly, for the motion estimation of a rigid body in a 3D space wheel encoders cannot be used. In such cases, IMU sensors could be sound but expensive options. Therefore, using a single camera and feature tracking methods to estimate motion of the camera in the 3D space has attracted the attention of researchers in the last decade. These methods are known as visual odometry methods. In this section, firstly rigid body transforms in 3D space will be presented, then cameras as projective sensors will be discussed and finally we introduce different methods known as 8-, 7- and 5-point methods to retrieve the motion parameters of a monocular camera.

2.2.1 Rigid Body Transformation

Given a Cartesian frame as a world frame, a rigid body transformation maps a subset of $\mathcal{R}^3$, such as $\mathcal{A}$, to another subset of $\mathcal{R}^3$, such as $\mathcal{B}$, in such a way that the distances of the points in two subsets are preserved. $h : \mathcal{A} \rightarrow \mathcal{B}$. From linear algebra, we know that affine transformations are defined by:

$$\begin{aligned} \mathbf{b} &= R\mathbf{a} + \mathbf{t} \\ \mathbf{a} \in \mathcal{A}, \mathbf{b} &\in \mathcal{B} \end{aligned} \tag{2.50}$$

where R is an orthonormal matrix such that $RR^T = R^T R = I$ and $\mathbf{t} = [t_x \; t_y \; t_z]^T$ is a translation vector. R is 3×3 matrix known as rotation matrix. A rotation matrix can be interpreted in several physical motions of which important ones

are as follows: First consecutive rotations about three orthogonal axis Z, Y and X with three angles ϕ, θ and ψ

$$R = R_z R_y R_x \tag{2.51}$$

$$\begin{aligned} R_z &= \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ R_y &= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \\ R_x &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \end{aligned} \tag{2.52}$$

Another physical interpretation of a rotation matrix is a rotation about a unit vector such as $\mathbf{u} = [u_x \ u_y \ u_z]^T$ with the angle θ:

$$R = \begin{bmatrix} c\theta + u_x^2(1-c\theta) & u_x u_y(1-c\theta) - u_z s\theta & u_x u_z(1-c\theta) + u_y s\theta \\ u_y u_x(1-c\theta) + u_z s\theta & c\theta + u_y^2(1-c\theta) & u_y u_z(1-c\theta) - u_x s\theta \\ u_z u_x(1-c\theta) - u_y s\theta & u_z u_y(1-c\theta) + u_x s\theta & c\theta + u_z^2(1-c\theta) \end{bmatrix}$$

where c and s represent cos and sin respectively. The important issue concerning a rotation matrix is that it has nine elements but it can be recovered from three parameters.

Assuming that a coordinate frame is attached to a rigid body, we denote the pose of the frame with respect to a global frame as $\{R|\mathbf{t}\}$, where R encodes the orientation of the frame and $\mathbf{t}$ is the position of the origin of the frame in the global frame. Now considering the point $\mathbf{p} = [p_x \ p_y \ p_z]^T$ in the global frame, we are interested to find the coordinate of the point in the rigid body frame: $\mathbf{p}' = [p'_x \ p'_y \ p'_z]^T$. It can be simply verified that $\mathbf{p}'$ is obtained as follows:

$$\mathbf{p}' = R^T(\mathbf{p} - \mathbf{t}) \tag{2.53}$$

2.2.2 Quaternion

Another way to encode the rotation of a rigid body is using a quaternion which looks simpler and computationally more efficient than a rotation matrix. Quaternions are an extension of imaginary numbers. A quaternion composes of a scalar part and a vector part as follows:

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{2.54}$$

where $q_0, ..., q_3 \in \mathcal{R}$, $i^2 = j^2 = k^2 = ijk = -1$ and $ij = k$, $jk = i$, $ki = j$ and $ji = -k$, $kj = -i$, $ik = -j$. If we want to show a pure vector or point such as $p = [p_x \; p_y \; p_z]^T$ by a quaternion, it should be written as follows: $p = p_x i + p_y j + p_z k$. Consequently, a rotation about a unit vector $u = u_x i + u_y j + u_z k$ with the angle θ can be encoded by a quaternion as follows:

$$e_{\mathbf{u},\theta} = \cos\frac{\theta}{2} + (u_x i + u_y j + u_z k)\sin\frac{\theta}{2} \tag{2.55}$$

With this definition, it can be proved that the rotation of a point such as $p = p_x i + p_y j + p_z k$ about a unit vector u can be obtained as follows:

$$p' = e_{u,\theta}\, p\, e^*_{u,\theta} \tag{2.56}$$

where $e^*_{u,\theta} = \cos\frac{\theta}{2} - (u_x i + u_y j + u_z k)\sin\frac{\theta}{2}$.

2.2.3 Pinhole Camera Model

Camera is a projective sensor which projects a point in 3D space such as $\mathbf{p} = [p_x \, p_y \, p_z]^T$ on the camera screen (retina) [Tsa87]. Looking at Fig. 2.19, we can see that:

$$\frac{x}{f} = \frac{p_x}{p_z} \tag{2.57}$$

$$\frac{y}{f} = \frac{p_y}{p_z} \tag{2.58}$$

where f is the focal length of the camera. The above model is known as the pinhole camera model. If we force $f = 1$, we obtain new coordinates for the projected points.

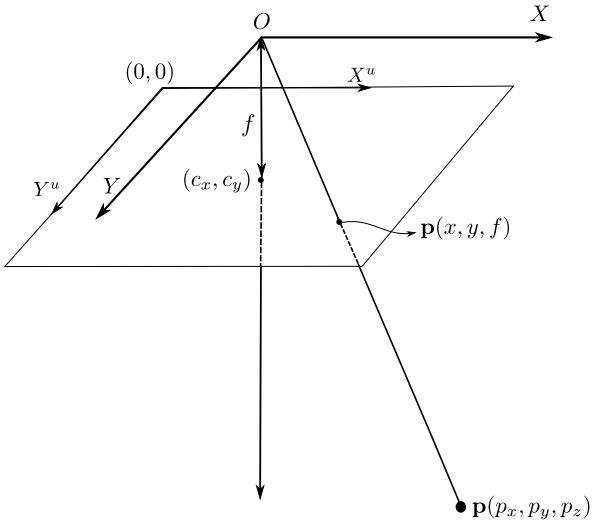


Figure 2.19: Pinhole camera model. X^u and Y^u are the axis of the uncalibrated camera coordinate system, X and Y are the axis of calibrated camera coordinate system, f is the focal length of the camera and (c_x, c_y) is the projection of the focal point on the camera screen.

In case of $f = 1$, the obtained model is known as the calibrated camera model. Nevertheless, what we get from a camera as measurements are the brightness or colors of pixels. The pixels are addressed based on their horizontal and vertical distances from the upper left corner of the screen. The units of the distances are in pixels. As a result, we need to transform the points in the pixel form into the normalized form such that we can benefit from the Eq. (2.58) for any geometrical analysis. In this regard, a procedure known as calibration should be conducted to obtain internal (intrinsic) parameters of cameras. The output of camera calibration will be a 3×3 matrix known as camera calibration matrix as follows:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.59}$$

f_x and f_y are the focal lengths of the camera in directions X and Y in pixels. The focal lengths might be different if the width and height of pixels are different. $[c_x\, c_y]^T$ is knows as the principal point showing the coordinates of the focal point of the camera on the uncalibrated camera screen. Given the camera calibration matrix, uncalibrated camera pixel coordinates can be transformed into normalized calibrated points as follows:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} x^u \\ y^u \\ 1 \end{bmatrix} \tag{2.60}$$

2.2.4 Single Camera Motion Recovery

Assuming two distinctive poses of a single camera in space and the two images captured by the camera at the poses, we intend to find the rigid body transformation $\{R^c, \mathbf{t}^c\}$ which moves the camera from the first pose to the second pose if a set of correspondent points between the two images are available. This problem is known in literature as ego-motion estimation.

A well-known method to estimate the transformation is to obtain a 3×3 matrix known as the essential matrix. Once an essential matrix is obtained, the rotation and translation parameters can be obtained from the singular value decomposition of the matrix. Higgins [LH81] and Tsai and Huang [TH84] approached the ego-motion estimation based on the essential matrix with almost similar methods. Given a point in the space which has the position $\mathbf{p} = [p_x\ p_y\ p_z]^T$ in the first camera coordinate and $\mathbf{p}' = [p'_x\ p'_y\ p'_z]$ in the second camera frame, we can write:

$$\mathbf{p}' = R\mathbf{p} + \mathbf{t} \tag{2.61}$$

where $R^T = R^c$ and $\mathbf{t}^c = \mathbf{t}$.

On the other hand, the point $\mathbf{p} = [p_x\ p_y\ p_z]^T$ is projected on a 2D point such as (x, y) on the retina of a calibrated camera as follows:

$$x = \frac{p_x}{p_z} \tag{2.62}$$

$$y = \frac{p_y}{p_z} \tag{2.63}$$

Considering the projection of a point in the space on the retina of a camera at two different positions which are two 2D points such as (x, y) and (x', y') and using Eq. (2.63), the following equation known as coplanarity equation is obtained:

$$[x'\ y'\ 1]E\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0 \tag{2.64}$$

where

$$E = \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{bmatrix} \tag{2.65}$$

which is known as the essential matrix in literature.

For an uncalibrated camera a similar equation can be obtained. By plugging Eq. (2.60) in Eq. (2.64), we have:

$$[x'^u\ y'^u\ 1]K^{-1^T}EK^{-1}\begin{bmatrix} x^u \\ y^u \\ 1 \end{bmatrix} = 0 \tag{2.66}$$

The term $F = K^{-1^T}EK^{-1}$ is known as the fundamental matrix in literature and widely used for analysis concerning uncalibrated cameras.

Nevertheless, for robust camera motion recovery, the intrinsic parameters of cameras should be known. Hence, we follow our discussion based on the essential matrix. Expansion of Eq. (2.64) results in:

$$[x'x,\ x'y,\ x',\ y'x,\ y'y,\ y',\ x,\ y,\ 1]\mathbf{e} = 0 \tag{2.67}$$

where $\mathbf{e} = [e_1 ... e_9]^T$.

If $e_9 \neq 0$ both sides of Eq. (2.64) can be divided by any nonzero factor or we can set e_9 to 1. Therefore, to determine E, the eight remaining elements should be calculated. Given eight corresponding points between two frames, an equation system consisting of eight equations is formed, by which the eight unknowns can be determined. To relax the necessity of $e_9 \neq 0$, an alternative solution is to form a homogeneous equation system and solve it using SVD as follows:

$$A\mathbf{e} = \mathbf{0} \tag{2.68}$$

where A is a $N \times 9$ ($N \geq 8$) matrix of which rows contain the coplanarity equation coefficients of each set of matched points (Eq. (2.67)). The equation system can be solved using the singular value decomposition under the constraint $\mathbf{e}^T\mathbf{e} = 1$. By obtaining the singular value decomposition of $A = U\Sigma V^T$, the column of V which corresponds to the smallest eigen value in Σ is the solution of Eq. (2.68). The above method is known as the 8-point method in literature [LH81].

Obviously, the eight equations should be independent in order that solving the equation system results in a unique solution. Tsai and Huang discussed that if seven points of the eight points lie on a plane, the equation system will be singular and E cannot be determined uniquely.

It can be verified that:

$$E = RT \tag{2.69}$$

where T is a skew symmetric matrix which includes only the translation parameters:

$$T = \begin{bmatrix} 0 & t_z & -t_y \\ -t_z & 0 & t_x \\ t_y & -t_x & 0 \end{bmatrix} \tag{2.70}$$

Using the singular value composition (SVD) of E, T and R can be calculated:

$$E = U\Sigma V^T \tag{2.71}$$

The matrix E should have two similar nonzero and one zero singular values. However, if there exist measurement noises, the values will be different. It means that the obtained E matrix does not satisfy Eq. (2.64). Tsai et al. showed that if the smallest singular value in Σ is forced to be zero then the left side of the equation would be minimized. Consequently, two solutions for R can be obtained as follows:

$$R = UWV^T; \quad W = \pm \begin{bmatrix} 0 & +1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.72}$$

and also there exist two solutions for the translation as follows:

$$T = VZV^T; \quad Z = \pm \begin{bmatrix} 0 & +1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.73}$$

Therefore, from an essential matrix four sets of solutions are obtained. Nonetheless, only one of them back projects the image points in front of the camera at both positions which will be the only possible solution.

Although the 8-point method is linear and fast, it has a poor performance if there exist minor measurement noises concerning the position of the correspondent points. Therefore, later some nonlinear iterative methods were introduced to calculate essential (fundamental) matrices [LDFP93]. The iterative methods are applicable if a good initial guess of the essential matrix elements is already available; otherwise, they get stuck in local minima.

Hartley [Har97] discussed that if the image coordinates are normalized based on the centroids of the matched points, the 8-point method performance would increase noticeably and even outperform some of the nonlinear methods. Looking at the experimental results shown by Hartley, it can be seen that still 8-point

method has poor performances for measurement noise of more than 0.1 pixel (for the image resolution 100×100).

One reason for the poor performance of the 8-point method is ignoring the dependencies of the essential matrix elements. Based on [Fau93], the dependencies of essential matrix elements can be represented as follows:

$$det(E) = 0 \tag{2.74}$$

$$EE^T E - \frac{1}{2} trace(EE^T)E = 0 \tag{2.75}$$

In [HZ04], Eq. (2.74) was used to come up with a method known as the 7-point method. To utilize Eq. (2.74), a homogeneous equation system including seven coplanarity equations is formed: $A\mathbf{e} = \mathbf{0}$. The matrix A has a null space spanned by two vectors such as $\mathbf{y}$ and $\mathbf{z}$. Thus, we have:

$$\mathbf{e} = y\mathbf{y} + \mathbf{z} \tag{2.76}$$

where $\mathbf{e} = [e_1 ... e_9]^T$. Plugging Eq. (2.76) in Eq. (2.74) results in a third order polynomial equation of y, which may have up to three real roots. It means that three valid essential matrices could explain the camera motion.

In [Nis04], Nister used Eq. (2.74) and Eq. (2.75) to propose his 5-point method. In this method, a homogeneous equation system based on five matched points is formed. The null space of the coefficient matrix of the equation system is spanned by four vectors such as $\mathbf{w}$, $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. Therefore, we can say:

$$\mathbf{e} = w\mathbf{w} + x\mathbf{x} + y\mathbf{y} + z\mathbf{z} \tag{2.77}$$

where w, x, y and z are unknown scalar values. The scalar values, however, can be calculated up to a common scale factor. Therefore, we can assume $w = 1$.

By plugging Eq. (2.77) into Eq. (2.74) and Eq. (2.75) a system of equations is obtained. Using Jordan elimination and removing x and y, finally, Nister achieved a polynomial equation of order ten of the parameter z. In [LH06], this

polynomial is obtained in another method which is not as efficient as the Nister's method.

By solving the polynomial and obtaining its real roots, the two other parameters x and y can also be obtained. Depending on the point arrangement, the number of valid solutions (the real roots of the polynomial) change. Obviously, it can be expected that there exist more than one real solution. After running several tests, Nister found that there were in average 2.74 solutions. Then, he argued if the five points were observed from three different views, almost in all cases a unique solution for the motion of the camera could be obtained.

The implementation of the 5-point method is difficult since in this algorithm symbolic processing is needed. Consequently, it will be also very slow. One shortcoming in both 5-point and 7-point methods is forcing one of the unknowns to one which is valid if they are not equal to zero; however, it cannot be guaranteed. Therefore, it can result in numerical errors if the coefficients are zero or near to zero.

2.3 Fusion

In this section, we will have a survey of different methods to fuse odometry data with the extrinsic sensor data. At first, typical planar motion and measurement models of a non-holonomic mobile robot is presented and then the fusion methods will be discussed.

2.3.1 Motion Model of a Planar Mobile Robot

As discussed in section 2.2, the instantaneous motion of a rigid body moving freely in the space can be presented by a rotation matrix and a translation vector. Nonetheless, a wheeled robot or vehicle can have only constrained motions; in such a way that it can only have motion in the direction of the current orientation of the robot. The motions of car like vehicles have one more constraint so that they cannot rotate without any translation.

The planar motion of a robot is a function of the rotation of its wheels. A simple form of a mobile robot is a two wheeled drive robot as shown in Fig. 2.20.

Figure 2.20: A two wheeled differential drive robot model Pioneer P3 DX (from [pio15]).

The pose of a robot on a plane is determined by three parameters x^R, y^R, θ^R. and the purpose of a motion model is to determine these three variables along the time given the torques or angular velocities of the wheels. If a motion model is derived based on the force or torque equations, it is called a dynamic model, and if the model is based on the angular velocities of the wheels, it is known as a kinematic model.

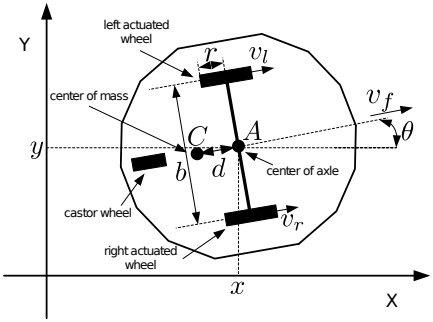


Figure 2.21: Parameters concerning a two wheeled mobile robot (from [pio15]).

The model which is usually used for the localization purpose is the kinematic model. Considering Fig. 2.21 the motion model of the robot can be written as follows:

$$\begin{aligned} \dot{x}_t^R &= u_{f,t} \cos\theta_t^R \\ \dot{y}_t^R &= u_{f,t} \sin\theta_t^R \\ \dot{\theta}_t^R &= u_{r,t} \end{aligned} \tag{2.78}$$

where $u_{f,t} = \frac{r}{2}\big(\omega_{r,t} + \omega_{l,t}\big)$, r is the radius of the wheels, $u_{r,t} = \frac{r}{b}\big(\omega_{r,t} - \omega_{l,t}\big)$, $\omega_{r,t}$ and $\omega_{l,t}$ are the angular speeds of right and left wheels. As in Fig. 2.21 can be seen x_t^R and y_t^R refer to the axle center of the robot. We denote the three variables of the robot pose in the vector $\mathbf{x}_t^R = [x_t^R\, y_t^R\, \theta_t^R]^T$.

The validity of this model can be verified easily under the condition that no slippage occurs during the movement of the robot.

In the case of four wheeled drive robots, since the robot should skid over its wheels in order to rotate, x_t^R and y_t^R refers to the center of the robot mass. With this consideration Eq. (2.78) can be extended for four wheeled robots.

The model represented by Eq. (2.78) is a continuous time model. To deal with the motion of a robot by digital computers, Eq. (2.78) should be discretized as follows:

$$\begin{aligned} x_t^R &= x_{t-1}^R + u_{f,t-1} \cos\theta_{t-1}^R \Delta t \\ y_t^R &= y_{t-1}^R + u_{f,t-1} \sin\theta_{t-1}^R \Delta t \\ \theta_t^R &= \theta_{t-1}^R + u_{r,t-1} \Delta t \end{aligned} \tag{2.79}$$

The discrete time model is obtained from the linear approximation of Eq. (2.78) about its operating point and it is valid if the sampling rate is high enough.

2.3.2 Measurement Model

Based on odometry data, we can calculate a robot pose at any time. However, due to various disturbances such as wheel slippages, sampling noise or imprecision concerning the center of a robot's mass, the pose calculation using the motion model can have an error at each time step, which accumulates along the time and gives rise to large errors in long expeditions. To address this problem, extrinsic sensors are utilized to provide relative measurements between the robot pose and landmark positions. We denote a landmark position in a 2D Cartesian coordinate system as $\mathbf{x}_j^L = [x_j^L \ y_j^L]^T$. These new measurements are fused with odometry data in order to confine the errors. The extrinsic sensors can be sonar, laser range finders or cameras. Range finder sensors have vast usages mostly for indoor missions of mobile robots. A range finder sensor can provide relative distances and angles between a robot and objects or landmarks (Fig. 2.22). Considering Fig. 2.22, the measurement model will be as follows:

$$\mathbf{h}_j(\mathbf{x}_t^R, \mathbf{x}_j^L) = \begin{bmatrix} d_{j,t} \\ \phi_{j,t} \end{bmatrix} = \begin{bmatrix} \sqrt{(x_j^L - x_t^R)^2 + (y_j^L - y_t^R)^2} \\ \tan^{-1} \frac{y_j^L - y_t^R}{x_j^L - x_t^R} - \theta_t^R \end{bmatrix} \tag{2.80}$$

where $j = 1...M$ and M is the number of landmarks.

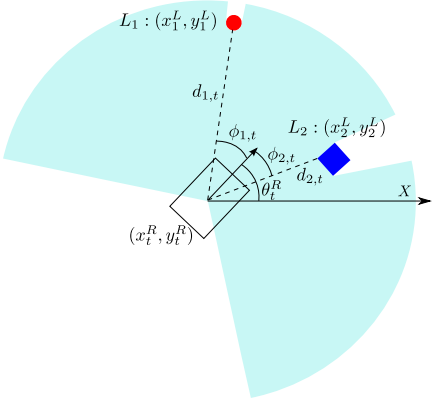


Figure 2.22: Measurements by range finder sensors.

2.3.3 Fusion

In [DNC+01], the SLAM problem is formulated in a state space scheme. Considering the motion and measurement models Eq. (2.79) and Eq. (2.80), a state space model can be formed as follows:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \\ \mathbf{z}_t &= \mathbf{h}(\mathbf{x}_t) + \mathbf{v}_t \end{aligned} \tag{2.81}$$

where $\mathbf{f} : \mathcal{R}^{5+2M} \rightarrow \mathcal{R}^{3+2M}$, $\mathbf{h} : \mathcal{R}^{3+2M} \rightarrow \mathcal{R}^{2M}$,

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{x}_t^R \\ \mathbf{x}_{1,t}^L \\ \vdots \\ \mathbf{x}_{M,t}^L \end{bmatrix} = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) = \begin{bmatrix} x_{t-1}^R + u_{f,t-1} \cos\theta_{t-1}^R \Delta t \\ y_{t-1}^R + u_{f,t-1} \sin\theta_{t-1}^R \Delta t \\ \theta_{t-1}^R + u_{r,t-1}\Delta t \\ \mathbf{x}_{1,t-1}^L \\ \vdots \\ \mathbf{x}_{M,t-1}^L \end{bmatrix},$$

$$\mathbf{h}(\mathbf{x}_t) = \begin{bmatrix} d_{1,t} & \phi_{1,t} & d_{2,t} & \phi_{2,t} & \dots & d_{M,t} & \phi_{M,t} \end{bmatrix}^T,$$

and $\mathbf{u}_t = [u_{f,t}\ u_{r,t}]^T$ is the input velocity vector is the measurement noise vector. The input vector can be written as the summation of measured velocities $\bar{\mathbf{u}}_t$ and their uncertainties $\tilde{\mathbf{u}}_t = [\tilde{u}_{f,t}\ \tilde{u}_{r,t}]^T$. $\tilde{\mathbf{u}}_t$ is assumed Gaussian ($\tilde{\mathbf{u}}_t \sim \mathcal{N}(\mathbf{0}, Q)$), where:

$$Q = \begin{bmatrix} \sigma_f^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix} \tag{2.82}$$

σ_f^2 and σ_r^2 are the variances of the uncertainties of the forward and angular velocities. The measurement noise vector is also considered as a Gaussian vector $\mathbf{v}_t = [v_{1,d}\ v_{1,\phi}\ \dots\ v_{M,d}\ v_{M,\phi}]^T \sim \mathcal{N}(\mathbf{0}, R)$, where:

$$R = \begin{bmatrix} \sigma_{1,d}^2 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sigma_{1,\phi}^2 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sigma_{2,d}^2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \sigma_{2,\phi}^2 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{M,d}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{M,\phi}^2 \end{bmatrix} \tag{2.83}$$

in which $\sigma_{j,d}^2$ and $\sigma_{j,\phi}^2$ are the variances of distance and bearing measurements. By forming the state space model, it can be inferred that the position estimation problem turns into a state estimation problem which is a classical problem in the control theory. The state estimation can be addressed by the Kalman filter method [Kal60] if both the motion and measurement models are linear. If the models are not linear, extended Kalman filters (EKF) are used, which apply linear approximations of the models at the operating point of the system. Using extended Kalman filters can be problematic if the models have severe nonlinearities. In this case, unscented Kalman filters (UKF) [JU04] or particle filters [GSS93] are proper substitutions. In the following, we first introduce the mentioned filters and then their special usages for the SLAM problem.

Kalman Filter

Given a linear state space model as follows:

$$\begin{aligned} \mathbf{x}_t &= F_{n\times n}\mathbf{x}_{t-1} + B_{n\times p}(\bar{\mathbf{u}}_{t-1} + \tilde{\mathbf{u}}_{t-1}) \\ \mathbf{z}_t &= H_{m\times n}\mathbf{x}_t + \mathbf{v}_t \end{aligned} \tag{2.84}$$

The goal is to have an optimal estimation of $\mathbf{x}_t$ given the input and measurement sequences $U_{t-1} = \{\mathbf{u}_{t-1}, ..., \mathbf{u}_0\}$ and $Z_t = \{\mathbf{z}_t, \mathbf{z}_t, ..., \mathbf{z}_0\}$:

$$p(\mathbf{x}_t | Z_t, U_{t-1}) \tag{2.85}$$

From the probability theory, we know that an affine transformation of a Gaussian vector is again Gaussian; hence, we have:

$$p(\mathbf{x}_t|Z_t, U_{t-1}) = \mathcal{N}(\bar{\mathbf{x}}_t, P_t) \tag{2.86}$$

P_t is the covariance of $\mathbf{x}_t$ and clearly the less its norm, the less uncertainty of $\mathbf{x}_t$. Consequently, an error vector can be defined as follows:

$$\mathbf{e}_t = \bar{\mathbf{x}}_t - \mathbf{x}_t \tag{2.87}$$

where $\mathbf{x}_t$ is the real state vector. By definition we have:

$$P_t = E[\mathbf{e}_t \mathbf{e}_t^T] \tag{2.88}$$

The optimal estimation of the state, however, can not be obtained directly and it should be solved recursively along the running of the system which includes prediction and update parts.

1. Prediction

$$\bar{\mathbf{x}}_{t|t-1} = E[\mathbf{x}_t|Z_{t-1}, U_{t-1}] = F\bar{\mathbf{x}}_{t-1|t-1} + B\bar{\mathbf{u}}_{t-1} \tag{2.89}$$

$$P_{t|t-1} = E[\mathbf{e}_t \mathbf{e}_t^T|Z_{t-1}] = FP_{t-1|t-1}F^T + BQB^T \tag{2.90}$$

2. Update

$$\bar{\mathbf{x}}_{t|t} = E[\mathbf{x}_t|Z_t] = \bar{\mathbf{x}}_{t|t-1} + P_{t|t-1}H^T S_t^{-1}(\mathbf{z}_t - H\bar{\mathbf{x}}_{t|t-1}) \tag{2.91}$$

$$P_{t|t} = E[\mathbf{e}_t \mathbf{e}_t^T|Z_t] = P_{t|t-1} - P_{t|t-1}H^T S_t^{-1} HP_{t|t-1} \tag{2.92}$$

in which S_t is known as the innovation covariance matrix and is calculated as follows:

$$S_t = HP_{t|t-1}H^T + R \tag{2.93}$$

Extended Kalman Filter

The derivation of Kalman filter formulations is based on the linear model shown in Eq. (2.84). Hence, we cannot directly use the equations for nonlinear systems, unless their Taylor approximation about the operating point of the system is used.

Given a nonlinear state space model as follows:

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \tag{2.94}$$

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{v}_t \tag{2.95}$$

we can derive the following linear approximation:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{f}(\bar{\mathbf{x}}_{t-1}, \bar{\mathbf{u}}_{t-1}) + F_{t-1}\tilde{\mathbf{x}}_{t-1} + B_{t-1}\bar{\mathbf{u}}_{t-1} + B_{t-1}\tilde{\mathbf{u}}_{t-1} \\ \mathbf{z}_t &= \mathbf{h}(\bar{\mathbf{x}}_t) + H_t\tilde{\mathbf{x}}_t \end{aligned} \tag{2.96}$$

where

$$F_{t-1} = \frac{\partial}{\partial \mathbf{x}}\mathbf{f}(.)\Big|_{\mathbf{x}=\bar{\mathbf{x}}_{t|t-1}} \; ; B_{t-1} = \frac{\partial}{\partial \mathbf{u}}\mathbf{f}(.)\Big|_{\mathbf{u}=\bar{\mathbf{u}}_{t-1}} \; ; H_t = \frac{\partial}{\partial \mathbf{x}}\mathbf{h}(.)\Big|_{\mathbf{x}=\bar{\mathbf{x}}_{t|t-1}}$$

EKF filters work well if the the nonlinearity of a system is not severe, otherwise the first order linearization will not be an appropriate approximation, which gives rise to the poor performance or even divergence of EKF filters. For such cases, particle filters or unscented Kalman filters can be applied.

Particle Filter

In [GSS93], Gordon et al. proposed a numerical method based on the random sampling of PDFs. The algorithm which is known as the particle filter has also

prediction-updated phases similar to the Kalman filter. Given the nonlinear state space model Eq. (2.81), the goal is to estimate the following PDF recursively:

$$p(\mathbf{x}_t|Z_t) \tag{2.97}$$

where $Z_t = \{\mathbf{z}_t, ..., \mathbf{z}_0\}$. The recursive algorithm is as follows:

1. **Predict:** Generate N_p samples (particles) from $p(\mathbf{x}_{t-1}|Z_{t-1})$ such as $X_{t-1} = \{\mathbf{x}_{t-1,1}, ..., \mathbf{x}_{t-1,N_p}\}$, and also generate N_p samples from $p(\tilde{\mathbf{u}})$ such as $\tilde{U}_{t-1} = \{\tilde{\mathbf{u}}_{t-1,1}, ..., \tilde{\mathbf{u}}_{t-1,N_p}\}$. Plug X_{t-1}, $\tilde{U}_{t-1}$ and the control input $\mathbf{u}_{t-1}$ into the motion equation Eq. (2.94) to obtain a set of predicted samples of the next state such as $X_{t|t-1} = \{\mathbf{x}_{t|t-1,1}, ..., \mathbf{x}_{t|t-1,N_p}\}$. Consequently, given the predicted particles and the measurement model, the predicted measurements are calculated:

$$\mathbf{z}_i = \mathbf{h}(\mathbf{x}_{t|t-1,i}); \quad i = 1...N_p \tag{2.98}$$

2. **Update:**

   As soon as the new measurement vector is obtained, the update PDF: $p(\mathbf{x}_t|Z_t)$, can be built up as follows:

$$p(\mathbf{x}_t|Z_t) = \sum_{i=1}^{N_p} w_i \delta(\mathbf{x}_t - \mathbf{x}_{t|t-1,i}) \tag{2.99}$$

   where w_i are the weights of the samples which are obtained as follows:

$$w_i = \frac{1}{c} e^{-\frac{1}{2}(\mathbf{z}_i - \mathbf{z})\Sigma_{\mathbf{v}}^{-1}(\mathbf{z}_i - \mathbf{z})^T} \tag{2.100}$$

   in which $c = \sum w_i$ is a normalization constant and $\Sigma_{\mathbf{v}}$ is the covariance matrix of the measurement noise.

Generally, the performance of a particle filter depends on the number of particles which should be dramatically increased with respect to the dimension of the estimated vector.

Unscented Kalman Filter

As stated before, the EKF filter has a poor performance if the system has a strong nonlinearity. In [JU04], Julier et al. have discussed the shortcoming in more details and offered a new filter based on the original formulation of the Kalman filter. The main idea is that from a PDF at each time step, some sample points (sigma points) are selected such that the first and the second order statistics of the samples are equal to the statistics of the PDF. Then the points are plugged into the motion and measurement models to achieve predicted terms. The sigma points are sampled deterministically. If the estimated vector $\mathbf{x}$ has a Gaussian distribution $\mathcal{N}(\bar{\mathbf{x}}, \Sigma_{\mathbf{x}})$ a proper set of points can be obtained at the border of a hyper ellipse presented by the following equation:

$$(\mathbf{x} - \bar{\mathbf{x}})\sqrt{N_{\mathbf{x}}\Sigma_{\mathbf{x}}}(\mathbf{x} - \bar{\mathbf{x}})^T = 0 \tag{2.101}$$

where $N_{\mathbf{x}}$ is the dimension of the vector $\mathbf{x}$. At this border, $2N_{\mathbf{x}}$ symmetric point are selected as follows:

$$\mathbf{x}_i = \bar{\mathbf{x}} + (\sqrt{N_{\mathbf{x}}\Sigma_{\mathbf{x}}})_i; \quad i = 1, ..., N_{\mathbf{x}} \tag{2.102}$$

$$w_i = \frac{1}{2N_{\mathbf{x}}} \tag{2.103}$$

$$\mathbf{x}_{i+N_{\mathbf{x}}} = \bar{\mathbf{x}} - (\sqrt{N_{\mathbf{x}}\Sigma_{\mathbf{x}}})_i \tag{2.104}$$

$$w_{i+N_{\mathbf{x}}} = \frac{1}{2N_{\mathbf{x}}} \tag{2.105}$$

where $(\sqrt{N_{\mathbf{x}}\Sigma_{\mathbf{x}}})_i$ represents the ith column of the matrix $\sqrt{N_{\mathbf{x}}\Sigma_{\mathbf{x}}}$

By introduction of sigma points, to address the state estimation problem, they have defined an augmented vector $\mathbf{x}_t^a$ and an augmented covariance matrix as follows:

$$\mathbf{x}_t^a = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \\ \mathbf{v}_{t+1} \end{bmatrix} \tag{2.106}$$

$$P_t^a = \begin{bmatrix} P_t & 0 & 0 \\ 0 & \Sigma_{\mathbf{u}_t} & 0 \\ 0 & 0 & \Sigma_{\mathbf{v}_t} \end{bmatrix} \tag{2.107}$$

and then rewrote the state space formulation as follows:

$$\mathbf{x}_t^a = \mathbf{f}(\mathbf{x}_{t-1}^a, \mathbf{u}_{t-1}) \tag{2.108}$$

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t^a) \tag{2.109}$$

Consequently, the prediction-update parts can be developed as follows:

1. **Prediction:** Form $\bar{\mathbf{x}}_{t-1|t-1}^a$ vector and $P_{t-1|t-1}^a$ covariance matrix and consequently extract $2N_{\mathbf{x}^a}$ from $p(\mathbf{x}_{t-1}^a|Z_{t-1})$ such as $X_{t-1}^a = \{\mathbf{x}_{t-1,1}^a, ..., \mathbf{x}_{t-1,N_{\mathbf{x}^a}}^a\}$ and plug them into the motion model and obtain a set of predicted points $X_{t|t-1}^a = \{\mathbf{x}_{t|t-1,1}^a, ..., \mathbf{x}_{t|t-1,N_{\mathbf{x}^a}}^a\}$. $N_{\mathbf{x}^a}$ is the size of $\mathbf{x}^a$. Based on the achieved points the predicted mean vector and covariance matrix can be obtained:

$$\bar{\mathbf{x}}_{t|t-1}^a = \sum_{i=0}^{2N_{\mathbf{x}^a}} w_i \, \mathbf{x}_{t|t-1,i}^a \tag{2.110}$$

$$P_{t|t-1}^a = \sum_{i=0}^{2N_{\mathbf{x}^a}} w_i \, [\mathbf{x}_{t|t-1,i}^a - \bar{\mathbf{x}}_{t|t-1}^a][\mathbf{x}_{t|t-1,i}^a - \bar{\mathbf{x}}_{t|t-1}^a]^T \tag{2.111}$$

   The predicted observation vectors are also as follows:

$$\mathbf{z}_{t|t-1,i} = \mathbf{h}(\mathbf{x}_{t|t-1,i}^a) \tag{2.112}$$

   and the predicted mean of the measurement vector is:

$$\bar{\mathbf{z}} = \sum_{i=0}^{2N_{\mathbf{x}^a}} w_i \mathbf{z}_{t|t-1,i} \tag{2.113}$$

2. **Update:** For the update part, we need to calculate the cross covariance matrix between the state vector and the measurement vector which is obtained as follows:

$$P^{\mathbf{xz}}_{t|t-1} = \sum_{i=0}^{2N_{\mathbf{x}^a}} w_i(\mathbf{x}_{t|t-1,i} - \bar{\mathbf{x}}_{t|t-1})(\mathbf{z}_{t|t-1,i} - \bar{\mathbf{z}}_{t|t-1})^T \tag{2.114}$$

The update equations can be written similar to the standard Kalman filter:

$$\mathbf{x}_{t|t} = \mathbf{x}_{t|t-1} + P^{\mathbf{xz}}_{t|t-1}(S_t)^{-1}(\mathbf{z}_t - \mathbf{z}_{t|t-1}) \tag{2.115}$$

$$P_{t|t} = P_{t|t-1} - P^{\mathbf{xz}}_{t|t-1}(S_t)^{-1}P^{\mathbf{xz},T}_{t|t-1} \tag{2.116}$$

In which S_t is an innovation covariance matrix and is calculated as follows:

$$S_t = \sum_{i=0}^{2N_{\mathbf{x}^a}} w_i(\mathbf{z}_{t|t-1,i} - \bar{\mathbf{z}}_{t|t-1})(\mathbf{z}_{t|t-1,i} - \bar{\mathbf{z}}_{t|t-1})^T \tag{2.117}$$

SLAM Using EKF

In [GN01], [DNC+01], [WDD02], the usage of EKF to address the SLAM problem has been discussed. Obviously, by having the motion and measurement models, we only need to calculate the linear approximations of the models. Considering Eq. (2.79) and Eq. (2.80), the parameters are obtained as follows:

$$F_{t-1} = \begin{bmatrix} 1 & 0 & -u_{f,t-1}\sin(\theta^R_{t-1}) & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & u_{f,t-1}\cos(\theta^R_{t-1}) & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}_{(3+2M)\times(3+2M)}$$

$$B_{t-1} = \begin{bmatrix} \cos\theta_{t-1}^{R} & 0 \\ \sin\theta_{t-1}^{R} & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 0 \end{bmatrix}_{(3+2M)\times 2}$$

$$H_t = [h_{ki}]_{2M\times(3+2M)} \tag{2.118}$$

where h_{ki} is the element of H at the row k and column i. The row $k = 2j$ of H contains the derivation of the distance measurement of j^{th} landmark with respect to all state variables and the row $k = 2j + 1$ contains the derivation of the bearing measurement of j^{th} landmark with respect to all state variables. The elements of H are determined as follows:

$$h_{2j,0} = -\frac{d_{x,j}}{\sqrt{d_{x,j}^2 + d_{y,j}^2}} \tag{2.119}$$

$$h_{2j,1} = -\frac{d_{y,j}}{\sqrt{d_{x,j}^2 + d_{y,j}^2}} \tag{2.120}$$

$$h_{2j,2} = 0 \tag{2.121}$$

$$h_{2j,3+2j} = \frac{d_{x,j}}{\sqrt{d_{x,j}^2 + d_{y,j}^2}} \tag{2.122}$$

$$h_{2j,4+2j} = \frac{d_{y,j}}{\sqrt{d_{x,j}^2 + d_{y,j}^2}} \tag{2.123}$$

$$h_{2j+1,0} = \frac{d_{y,j}}{d_{x,j}^2 + d_{y,j}^2} \tag{2.124}$$

$$h_{2j+1,1} = \frac{d_{x,j}}{d_{x,j}^2 + d_{y,j}^2} \tag{2.125}$$

$$h_{2j+1,2} = -1 \tag{2.126}$$

$$h_{2j+1,3+2j} = -\frac{d_{y,j}}{d_{x,j}^2 + d_{y,j}^2} \tag{2.127}$$

$$h_{2j+1,4+2j} = -\frac{d_{x,j}}{d_{x,j}^2 + d_{y,j}^2} \tag{2.128}$$

where $d_{x,j} = x_j^L - x_t^R$ and $d_{y,j} = y_j^L - y_t^R$.

Fast SLAM

Using the EKF filter is straightforward, however, it suffers from two shortcomings. First, increasing the complexity of the algorithm quadratically with respect to the number of landmarks. Second, the violation of the Gaussian assumption if input noise or measurement noise is noticeable. To address these problems, Montemerlo et al. have proposed a method known as Fast-SLAM [MTKW02]. The method is based on a variant of particle filters called the Rao-Blackwellized particle filter [CR96]. They also augmented the problem of feature association in the SLAM problem. In a probabilistic term, the SLAM problem can be stated as follows:

$$p(X_t^R, X^L | Z_t, U_t, n_t) \tag{2.129}$$

where $X_t^R = \{\mathbf{x}_t^R, \mathbf{x}_{t-1}^R, ..., \mathbf{x}_0^R\}$ is the set of all the robot poses by the time t (path), $X^L = \{\mathbf{x}_1^L, \mathbf{x}_2^L, ..., \mathbf{x}_M^L\}$ denotes the set of landmark positions, $Z_t = \{\mathbf{z}_t, \mathbf{z}_{t-1}, ..., \mathbf{z}_0\}$ is the set of all measurements by the current time, $U_t = \{\mathbf{u}_t, \mathbf{u}_{t-1}, ..., \mathbf{u}_0\}$ is the set of all control inputs by the current time and $n_t \subset \{1, ..., M\}$ is the set of indexes of the observed landmarks at time t.

Since the positions of the landmarks are independent from each other, Eq. (2.129) can be decomposed into the following form:

$$p(X_t^R, X^L|Z_t, U_t, n_t) = p(X_t^R|Z_t, U_t, n_t) \prod_{j=1}^{M} p(\mathbf{x}_j^L|X_t^R, Z_t, U_t, n_t) \quad (2.130)$$

The term $p(X_t^R|Z_t, U_t, n_t)$ is handled by a particle filter and the terms $p(\mathbf{x}_j^L|X_t^R, Z_t, U_t, n_t)$ are handled by EKF filters. Hence, each particle includes a path and M EKF filters:

$$\{X_{t,i}^R, \mathbf{x}_{1,i}^L, ..., \mathbf{x}_{M,i}^L\} \quad (2.131)$$

Assuming that the robot pose at the time 0 has the prior distribution $p(\mathbf{x}_0^R)$, and at this time M landmarks are observed, then for each sample (particle) $\mathbf{x}_{0,i}^R$ from the prior PDF, M distinct Gaussian distributions for the M landmarks can be initialized. The initialization is done using the measurement model. From Eq. (2.80), the following equation can be derived:

$$\mathbf{x}_{j,i}^L = \mathbf{g}(\mathbf{z}_0, \mathbf{x}_{0,i}^R, \mathbf{v}_0) \quad (2.132)$$

Using the linear terms of the Taylor expansion of Eq. (2.132), we have:

$$\mathbf{x}_{j,i}^L = \bar{\mathbf{x}}_{j,i}^L + \frac{\partial \mathbf{x}_{j,i}^L}{\partial \mathbf{v}_t}\Big|_{\mathbf{x}_{j,i}^L = \bar{\mathbf{x}}_{j,i}^L} \mathbf{v}_t \quad (2.133)$$

where $\bar{\mathbf{x}}_{j,i}^L = \mathbf{g}(\mathbf{z}_0, \mathbf{x}_{0,i}^R, \mathbf{0})$.

Assuming $\mathbf{v}_t \sim \mathcal{N}(0, R)$, $\mathbf{x}_{j,i}^L$ has a Gaussian distribution such as $\mathcal{N}(\bar{\mathbf{x}}_{j,i}^L, DRD^T)$, where $D = \frac{\partial \mathbf{x}_{j,i}^L}{\partial \mathbf{v}_t}\Big|_{\mathbf{x}_{j,i}^L = \bar{\mathbf{x}}_{j,i}^L}$. Obviously, the above initialization method can be extended for any time instance for any new landmark.

After the initialization step the following prediction-update phases can be derived:

1. **Prediction:** From the PDF $p(X_t^R|Z_t, U_t, n_t)$, N_p particles $X_{t,1}^R, ..., X_{t,N}^R$ are drawn based on their correspondent weights $\{w_1, ..., w_N\}$. As discussed

before, M EKF filters associated to the M landmarks are also attached to each particle. Therefore, a particle is a set as follows:

$$\{X^R_{t,i}, \mathbf{x}^L_{1,t,i}, ..., \mathbf{x}^L_{M,t,i}\} \tag{2.134}$$

where $\mathbf{x}^L_{j,t,i} \sim \mathcal{N}(\bar{\mathbf{x}}^L_{j,t,i}, P_{j,t,i})$. The index t is also added for landmark positions to represent the dynamic of the recursive algorithm.

For each drawn particle a predicted position $\mathbf{x}^R_{t|t-1,i}$ is obtained by using the motion model. Consequently, for each predicted position, M predicted measurements are calculated:

$$\bar{\mathbf{z}}_{t,j,i} = \mathbf{h}_j(\mathbf{x}^R_{t|t-1,i}, \bar{\mathbf{x}}^L_{j,t|t-1,i}) \tag{2.135}$$

where $\mathbf{h}_j(.,.) = [d_j, \phi_j]^T$ is the measurement model for the j^{th} landmark. $\bar{\mathbf{z}}_{t,j,i}$ is the predicted measurement calculated based on the i^{th} particle and j^{th} landmark at time t: Since the landmarks are assumed static, predictions of landmark uncertainties are skipped.

2. **Update:** As soon as, new measurements are obtained, the importance weights of the particles can be calculated as follows:

$$w_i = \frac{1}{c} e^{-\frac{1}{2}(\bar{\mathbf{z}}_{t,i} - \mathbf{z}_t)^T R^{-1} (\bar{\mathbf{z}}_{t,i} - \mathbf{z}_t)} \tag{2.136}$$

where $\bar{\mathbf{z}}_{t,i}$ is the vector of all predicted measurements for the i^{th} particle and $\mathbf{z}_t$ is the vector of all real measurements at time t. Each EKF filter attached to each particle is updated as follows:

$$\begin{aligned} \bar{\mathbf{x}}^L_{j,t|t,i} &= \bar{\mathbf{x}}^L_{j,t|t-1,i} + P_{j,t|t-1,i} H^T_{j,i} (S_{j,t,i})^{-1} \big(\mathbf{z}_j - \mathbf{h}_j\big(\mathbf{x}^R_{t|t-1,i}, \bar{\mathbf{x}}^L_{j,t|t-1,i}\big)\big) \\ P_{j,t|t,i} &= P_{j,t|t-1,i} - P_{j,t|t-1,i} H^T_{j,i} (S_{j,t,i})^{-1} H_{j,i} P_{j,t|t-1,i} \end{aligned}$$

where

$$H_{j,i} = \frac{\partial \mathbf{h}_j}{\partial \mathbf{x}_{j,t|t-1,i}}\Big|_{\mathbf{x}_{j,t|t-1,i} = \bar{\mathbf{x}}_{j,t^1 t-1,i}}, \ \bar{\mathbf{x}}_{j,t|t-1,i} = \begin{bmatrix} \mathbf{x}^R_{t|t-1,i} \\ \mathbf{x}^L_{j,t|t-1,i} \end{bmatrix},$$

$\mathbf{z}_j$ is the real measurement corresponding to the j^{th} landmark and $S_{j,t,i} = H_{j,i}P_{j,t|t-1,i}H_{j,i}^T + R$.

Since the fast SLAM method is based on the particle filter method, the number of particles is an issue which should be increased as the uncertainty of odometry data increases.

Cost Optimization Based Methods

The SLAM problem can also be considered as an optimization problem. In this regard, a cost function is defined over the differences between the real measurements and the expected measurements as follows:

$$C = \sum_{t=0}^{N_r} \sum_{j=1}^{M} \left(\mathbf{z}_{t,j} - \mathbf{h}_j(\mathbf{x}_t^R, \mathbf{x}_j^L)\right)^T \left(\mathbf{z}_{t,j} - \mathbf{h}_j(\mathbf{x}_t^R, \mathbf{x}_j^L)\right) \tag{2.137}$$

where N_r is the number of all robot poses and M is the number of all landmarks. Consequently, the parameters of the SLAM problem (robot and landmark positions) can be obtained by the minimization of the cost function.

Obviously, in real scenarios, each robot pose is only related to some landmarks; therefore, the optimization process can be skipped at many combinations of robot and landmark positions. This property is known as sparsity. Since the measurement models are typically nonlinear, Eq. (2.137) should be minimized iteratively.

A well known method to minimize such cost functions is the Gauss-Newton algorithm. To use this algorithm, firstly, a vector of all parameters should be defined such as: $\mathbf{x} = [\mathbf{x}_0^{R\,T}, ..., \mathbf{x}_{N_r}^{R\,T}, \mathbf{x}_1^{L\,T}, ..., \mathbf{x}_M^{L\,T}]^T$. Then Eq. (2.137) can be rewritten as follows:

$$C(\mathbf{x}) = \sum_{t=0}^{N_r} \sum_{j=1}^{M} \left(\mathbf{z}_{j,t} - \mathbf{h}_{j,t}(\mathbf{x})\right)^T \left(\mathbf{z}_{j,t} - \mathbf{h}_{j,t}(\mathbf{x})\right) \tag{2.138}$$

where $\mathbf{h}_{j,t}(\mathbf{x}) = \mathbf{h}_j(\mathbf{x}_t^R, \mathbf{x}_j^L)$ is the measurement model for j^{th} landmark observed at the robot pose $\mathbf{x}_t^R$. As the measurement models are nonlinear, the minimization should be done using iterative methods such as the Gauss-Newton method.

To this end, to find the necessary rule to update parameters in each iteration, the sensitivity of $\mathbf{h}_{j,t}$ with respect to $\mathbf{x}_t$ should be obtained:

$$\mathbf{h}_{j,t}(\mathbf{x}_t + \delta) \approx \mathbf{h}_{j,t}(\mathbf{x}_t) + J_{j,t}\delta \tag{2.139}$$

where $J_{j,t} = \frac{\partial \mathbf{h}_{j,t}}{\partial \mathbf{x}_t}$.

Therefore, we have:

$$C(\mathbf{x}+\delta) = \sum_{t=0}^{N_r}\sum_{j=0}^{M}(\mathbf{z}_{j,t} - \mathbf{h}_{j,t}(\mathbf{x}) + J_{j,t}\delta)^T(\mathbf{z}_{j,t} - \mathbf{h}_{j,t}(\mathbf{x}) + J_{j,t}\delta) \tag{2.140}$$

Eq. (2.140) can be written in a vector form as follows:

$$C(\mathbf{x}+\delta) = [\mathbf{z} - \mathbf{h}(\mathbf{x}) - J\delta]^T[\mathbf{z} - \mathbf{h}(\mathbf{x}) - J\delta] \tag{2.141}$$

where $\mathbf{z}$, $\mathbf{h}$ and J stack $\mathbf{z}_{j,t}$, $\mathbf{h}_{j,t}$ and $J_{j,t}$ respectively. Now, the goal is to find δ to minimize C. For this purpose, the derivation of Eq. (2.140) with respect to δ should be set to zero, which results in the following equation:

$$(J^T J)\delta = J^T[\mathbf{z} - \mathbf{h}(\mathbf{x})] \tag{2.142}$$

By solving the above equation, the modification factor δ in each iteration can be obtained. Levenberg and Marquardt modified Eq. (2.142) by adding a damping factor such as $\lambda \geq 0$ to regularize the convergence rate:

$$(J^T J + \lambda I)\delta = J^T[\mathbf{z} - \mathbf{h}(\mathbf{x})] \tag{2.143}$$

where I is an identity matrix. λ takes greater values in each iteration if C decreases slowly. optimization based methods have two major drawbacks: Firstly, they require a good initial guess of the parameters, otherwise, they get stuck in local minima. Secondly, they are computationally expensive. Obviously, to deal with the first problem, high quality odometry data is required. To alleviate the second issue, some methods have been offered that we review in the following.

Graph SLAM

The first method known as GraphSLAM was proposed by Thrun and Montemerlo [TM05]. They formulated the SLAM problem as a graph of which nodes are the robot and the landmark positions. The odometry data and measurements also constitute the edges of the graph. With this definition, the SLAM problem turns into a graph optimization problem which can be realized with the optimization based methods. The cost function in this case is defined as follows:

$$C = \sum_t \big(\mathbf{x}_t^R - \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})\big)^T Q_t^{-1} \big(\mathbf{x}_t^R - \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})\big) + \sum_t \sum_j \big(\mathbf{z}_{j,t} - \mathbf{h}_j(\mathbf{x}_t^R, \mathbf{x}_{j,t}^L)\big)^T R_t^{-1} \big(\mathbf{z}_{j,t} - \mathbf{h}_j(\mathbf{x}_t^R, \mathbf{x}_{j,t}^L)\big) \tag{2.144}$$

If $\mathbf{y} = [\mathbf{x}_0^{R\,T}, ..., \mathbf{x}_N^{R\,T}, \mathbf{x}_1^{L\,T}, ..., \mathbf{x}_M^{L\,T}]^T$ is Gaussian with the covariance matrix Σ, they followed their formulation by defining an information matrix $\Omega = \Sigma^{-1}$ and an information vector $\xi = \Omega\mathbf{y}$. Most of the off-diagonal elements of Σ are zero except the elements which link two consecutive robot poses or the elements which link a robot pose to a landmark position which is observed at the robot pose.

The graphSLAM can be summarized in four steps:

1. **Initialization:** In this step, the information vector and information matrix are initialized. For the initialization of the information matrix, firstly, using the Taylor expansion, Eq. (2.144) is rewritten as follows:

$$C = \sum_t \Big(\mathbf{x}_t^R - \mathbf{f}(\bar{\mathbf{x}}_{t-1}^R, \bar{\mathbf{u}}_{t-1}) - F_{t-1}(\mathbf{x}_{t-1}^R - \bar{\mathbf{x}}_{t-1}^R)\Big)^T Q_t^{-1} \Big(\mathbf{x}_t^R - \mathbf{f}(\bar{\mathbf{x}}_{t-1}^R, \mathbf{u}_{t-1}) - F_{t-1}(\mathbf{x}_{t-1}^R - \bar{\mathbf{x}}_{t-1}^R)\Big) + \sum_t \sum_j (\mathbf{z}_{j,t} - \mathbf{h}_j(\bar{\mathbf{y}}_t) - H_{j,t}(\mathbf{y}_t - \bar{\mathbf{y}}_t))^T R_t^{-1} (\mathbf{z}_{j,t} - \mathbf{h}_j(\bar{\mathbf{y}}_t) - H_{j,t}(\mathbf{y}_t - \bar{\mathbf{y}}_t)) \tag{2.145}$$

where $\mathbf{y}_t = [\mathbf{x}_t^{R\,T}, \mathbf{x}_1^{L\,T}, ..., \mathbf{x}_M^{L\,T}]^T$, $\mathbf{h}_j(\mathbf{y}_t) = \mathbf{h}_j(\mathbf{x}_t^{R\,T}, \mathbf{x}_1^{L\,T})$, $F_{t-1} = \frac{\partial \mathbf{f}}{\mathbf{x}_{t-1}^R}$ and $H_{j,t} = \frac{\partial \mathbf{h}_j(\mathbf{y}_t)}{\mathbf{y}_t}$.

Eq. (2.145) can be rewritten as follows:

$$\begin{aligned} C = & \sum_t {\mathbf{x}^R}^T_{t-1:t} \begin{bmatrix} I \\ -F_t \end{bmatrix} Q_t^{-1} \begin{bmatrix} I & -F_t \end{bmatrix} \mathbf{x}^R_{t-1:t} + \\ & {\mathbf{x}^R}^T_{t-1:t} \begin{bmatrix} I \\ -F_t \end{bmatrix} Q_t^{-1} [\mathbf{f}(\mathbf{u}_{t-1}, \bar{\mathbf{x}}_{t-1}) + F_{t-1}\bar{\mathbf{x}}_{t-1}] \\ & \sum_t \sum_j \mathbf{y}_t^T H_{j,t}^T R_t^{-1} H_{j,t} \mathbf{y}_t + \\ & \mathbf{y}_t^T H_{j,t}^T R_t^{-1} [\mathbf{z}_{j,t} - \mathbf{h}(\bar{\mathbf{y}}_t) - H_{j,t}\bar{\mathbf{y}}_t] \end{aligned} \tag{2.146}$$

where $\mathbf{x}^R_{t-1:t}$ denotes a vector which consists of the robot poses $\mathbf{x}^R_t$ and $\mathbf{x}^R_{t-1}$. By defining a vector which contains all robot and landmark positions such as $\mathbf{y} = [\mathbf{x}^R_0, ..., \mathbf{x}^R_t, \mathbf{x}^L_1, ..., \mathbf{x}^R_M]^T$ we can write Eq. (2.146) as follows:

$$C = \text{const.} + \mathbf{y}^T \Omega \mathbf{y} - 2\mathbf{y}^T \xi \tag{2.147}$$

where const. is a constant, Ω and ξ accumulate the coefficients of quadratic and linear terms respectively. Clearly, the information vector ξ and the information matrix Ω can be initialized based on Eq. (2.146) and Eq. (2.147).

2. **Marginalizing out landmark positions:** Thrun et al. used the marginalization rule for a random Gaussian vector to obtain the optimal estimation for the robot path. They proved that if $p(\mathbf{x}, \mathbf{y})$ is a joint Gaussian distribution with the following information vector and matrix:

$$\bar{\xi} = \begin{bmatrix} \bar{\xi}_{\mathbf{x}} \\ \bar{\xi}_{\mathbf{y}} \end{bmatrix}; \quad \Omega = \begin{bmatrix} \Omega_{\mathbf{xx}} & \Omega_{\mathbf{xy}} \\ \Omega_{\mathbf{yx}} & \Omega_{\mathbf{yy}} \end{bmatrix} \tag{2.148}$$

Then $p(\mathbf{x})$ will be also Gaussian with the following information vector and matrix:

$$\begin{aligned} \bar{\xi}_{\mathbf{x},marg} &= \bar{\xi}_{\mathbf{x}} - \Omega_{\mathbf{xy}} \Omega_{\mathbf{yy}}^{-1} \bar{\xi}_{\mathbf{y}} \\ \Omega_{\mathbf{xx},marg} &= \Omega_{\mathbf{xx}} - \Omega_{\mathbf{xy}} \Omega_{\mathbf{yy}}^{-1} \Omega_{\mathbf{yx}} \end{aligned} \tag{2.149}$$

Using the above rule, the landmark positions can be marginalized out to obtain a vector containing only robot poses and its corresponding information matrix as follows:

$$\begin{aligned} \bar{\xi}_{\mathbf{x}^R_{0:t},marg} &= \bar{\xi}_{\mathbf{x}^R_{0:t}} - \sum_j \Omega_{\mathbf{x}^R_{0:t},\mathbf{x}^L_j} \Omega^{-1}_{\mathbf{x}^L_j,\mathbf{x}^L_j} \xi_{\mathbf{x}^L_j} \\ \Omega_{\mathbf{x}^R_{0:t},\mathbf{x}^R_{0:t},marg} &= \Omega_{\mathbf{x}^R_{0:t},\mathbf{x}^R_{0:t}} - \sum_j \Omega_{\mathbf{x}^R_{0:t},\mathbf{x}^L_j} \Omega^{-1}_{\mathbf{x}^L_j,\mathbf{x}^L_j} \Omega_{\mathbf{x}^L_j,\mathbf{x}^R_{0:t}} \end{aligned} \quad (2.150)$$

3. **Calculating the optimal path:** After achieving the marginalized information vector and matrix, the optimal path and the related covariance matrix can be obtained:

$$\begin{aligned} \bar{\mathbf{x}}^R_{0:t} &= \Sigma_{0:t}\, \bar{\xi}_{\mathbf{x}^R_{0:t},marg} \\ \Sigma_{0:t} &= \Omega^{-1}_{\mathbf{x}^R_{0:t},\mathbf{x}^R_{0:t},marg} \end{aligned} \quad (2.151)$$

4. **Rebuilding the map:** By obtaining the best estimation of the robot poses, the map can be rebuilt using the inversion of the measurement model and the optimal estimation of the robot poses.

The graph SLAM method is based on the marginalization technique which is relatively computationally expensive. Additionally, it relies on the linear approximations which are insufficient if the models have severe nonlinearities. These problems are addressed by the $\sqrt{SAM}$ method to some extent.

Square Root Smoothing and Mapping ($\sqrt{SAM}$)

Dellaert et al. proposed in [DK06] an optimization based method to use the sparsity nature of the SLAM problem to optimize the related cost function. In the $\sqrt{SAM}$ method the cost function Eq. (2.137) is rewritten as follows:

$$C = \sum_{t=1}^{N_r} ||\mathbf{f}(\mathbf{x}^R_{t-1}, \mathbf{u}_{t-1}) - \mathbf{x}^R_t||^2_{P_t} + \sum_{i=1}^{N_\mathbf{z}} ||\mathbf{h}_j(\mathbf{x}^R_{t,i}, \mathbf{x}^L_{j,i}) - \mathbf{z}_i||^2_{V_i} \quad (2.152)$$

where $N_{\mathbf{z}}$ is the number of all measurements gathered at all robot poses. $||.||_P^2$ and $||.||_V^2$ represent the Mahalanobis distances. In Eq. (2.152), the association of a robot pose at time t and the j^{th} landmark position with the i^{th} measurement are denoted with the indexes (t,i) and (i,j) respectively. By linearizing Eq. (2.152) we have:

$$C = \sum_{t=1}^{N_r} ||F_{t-1}\,\delta_{\mathbf{x}_{t-1}^R} + G_t\,\delta_{\mathbf{x}_t^R} - \mathbf{a}_t||_{P_t}^2 + \sum_{i=1}^{N_{\mathbf{z}}} ||H_{i,t}\,\delta_{\mathbf{x}_{t,i}^R} + J_{i,j}\,\delta_{\mathbf{x}_{j,i}^L} - \mathbf{c}_i||_{V_i}^2 \tag{2.153}$$

where $j = 1...M$,

$$F_{t-1} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}_{t-1}}\Big|_{\bar{\mathbf{x}}_{t-1}^R}, \quad H_{i,t} = \frac{\partial \mathbf{h}_j}{\partial \mathbf{x}_{t,i}^R}\Big|_{\bar{\mathbf{x}}_{t,i}^R}, \quad J_{j,i} = \frac{\partial \mathbf{h}_j}{\partial \mathbf{x}_{j,i}^L}\Big|_{\bar{\mathbf{x}}_{j,i}^L}, \quad G = -I_{3\times 3},$$
$$\mathbf{a}_t = \bar{\mathbf{x}}_t^R - \mathbf{f}(\bar{\mathbf{x}}_{t-1}^R, \mathbf{u}_{t-1}), \quad \mathbf{c}_i = \mathbf{z}_i - \mathbf{h}_j(\bar{\mathbf{x}}_{t,i}^R, \bar{\mathbf{x}}_{j,i}^L).$$

To dispose of P_t and V_i in Eq. (2.153), the terms F_t , G_t and $\mathbf{a}_t$ should be multiplied by $(P_t^{-1/2})^T$. The terms $H_{i,t}$, $J_{i,j}$ and $\mathbf{c}_i$ should be multiplied by ${V_i^{-1/2}}^T$. Consequently, the following least square equation is obtained:

$$A\delta = \mathbf{b} \tag{2.154}$$

where δ concatenates $\delta_{\mathbf{x}_0^R}, ..., \delta_{\mathbf{x}_N^R}$ and $\delta_{\mathbf{x}_1^L}, ..., \delta_{\mathbf{x}_M^L}$. The structure of A can be presented for $N_r = 3$, $N_{\mathbf{z}} = 4$ and $M = 2$ as follows:

$$A = \begin{bmatrix} G_1 & & & & & \\ F_1 & G_2 & & & & \\ & F_2 & G_3 & & & \\ H_{1,1} & & & J_{1,1} & & \\ H_{2,1} & & & & J_{2,2} & \\ & H_{3,2} & & J_{3,1} & & \\ & & H_{4,3} & & J_{4,2} & \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \\ \mathbf{c}_4 \end{bmatrix} \tag{2.155}$$

In Eq. (2.155), the elements in A which are not presented are filled with zeros. The least squares solution of Eq. (2.154) is given by the following equation:

$$A^T A\delta = A^T \mathbf{b} \tag{2.156}$$

$\Omega = A^T A$ is an information matrix. The inversion of the matrix is the most computationally expensive part to solve the above equation. Dellaert et al. used the Cholesky factorization of $\Omega = R^T R$ where R is an upper triangular matrix. Thanks to the sparsity of the information matrix, Ω the Cholesky factorization is carried out relatively fast and Eq. (2.156) can be solved efficiently.

The above discussed optimization based methods are mostly used for offline enhancement of path estimation after the detection of loop closure. It means if the robot after a run comes back to the starting point; while the estimated robot pose has some error, the error can be minimized based on the weighs of the edges in the provided graph.

2.3.4 Bearing Only SLAM

The methods discussed before had the assumptions that the uncertainties of the landmarks can be modeled by Gaussian distributions. FastSLAM uses a particle filter to model the robot pose uncertainties, but still uses Gaussian assumptions for the landmark positions.

A relatively new SLAM problem is bearing only SLAM in which only bearing measurement sensors are used. A well known example of such sensors are monocular cameras. Thanks to various existing optical feature tracking algorithms, it is possible to extract several features from one frame and track them in next frames. Additionally, compared to range finder sensors, cameras are much cheaper. Bearing only SLAM, however, is a much more complex problem than the common SLAM problem in which both range and bearing measurements are available. The main challenge of the bearing only SLAM is large uncertainties of landmark positions which cannot be modeled by Gaussian distributions. As can be seen in Fig. 2.23, given the robot pose $\mathbf{x}_t^R$ and the bearing measurement $\phi_{j,t}$ with a Gaussian uncertainty, the support of the landmark position uncertainty will be a triangle with an infinite length. Nevertheless, if the robot pose has

also uncertainties, the distribution of the landmark position has a trapezoidal support. Obviously, such uncertainties cannot be modeled by a single Gaussian distribution.

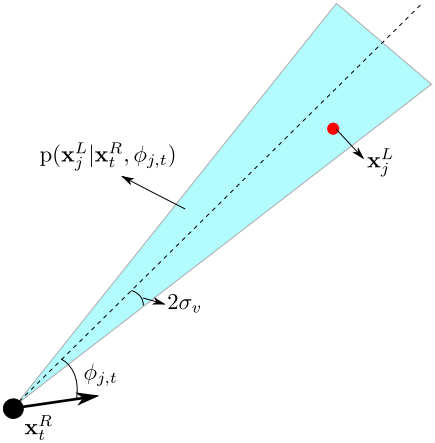


Figure 2.23: Initialization of a landmark uncertainty given a robot pose ($\mathbf{x}_t^R$) and a bearing measurement ($\phi_{j,t}$).

Delayed Bearing Only SLAM

In [Bai03], a delayed method is proposed so that the initialization of a landmark in the Gaussian form becomes possible. They used two different observations of a landmark to have a rough estimation of its position (Fig. 2.24). This method gives rise to increasing the uncertainty of the robot pose until the time that the second proper observation could be achieved. Furthermore, in real scenarios, it is possible to miss the track of many of the features before the second useful observation can be made. Therefore, un-delayed methods have attracted the attention of the researchers later.

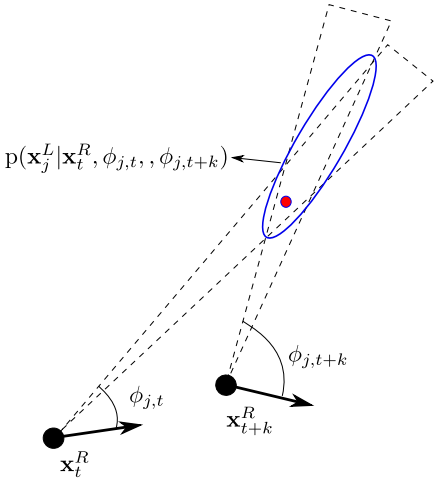


Figure 2.24: Initialization of a landmark uncertainty given two robot poses and two measurements.

Gaussian Sum Filter Method

In [KD04], Kwok and Dissanayake proposed the first un-delayed method in which multi-hypotheses of depths for each landmark are applied. Assuming that a landmark is expected to exist at a maximum range, a number of Gaussian distributions are placed along the line at which the landmark is observed (Fig. 2.25).

In this method, each hypothesis is assumed as a new landmark in the state vector. Hence, if we assume two hypotheses for M landmarks, $2M$ landmarks should be initialized in the state vector, resulting in a complexity of $\mathcal{O}((2M)^2)$. By motion of the robot and gathering new measurements of landmarks, most of the wrong hypotheses result in high predicted measurement errors and based on the errors theses hypothesizes can be removed from the state vector. In addition to the high complexity of the method, the method cannot deal with the ambiguities when two different landmarks are observed at the first time at almost the same direction.

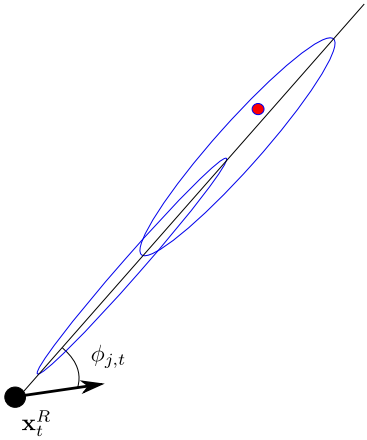


Figure 2.25: Landmark initialization using the multi hypothesis method.

To address the ambiguity problems, [KHH+05] Kwok et al. used Gaussian sum filters (GSF) to model the non-Gaussian uncertainties of the landmarks more intuitively. The implementation of GSF can be done using M^{N_g} parallel EKF filters where N_g is the number of Gaussian distributions used to model the uncertainty of each landmark. Therefore, the PDF of the state vector can be written as follows:

$$p(\mathbf{x}_t) = \sum_{i=1}^{M^{N_g}} w_i \mathcal{N}(\bar{\mathbf{x}}_{t,i}, P_{t,i}) \tag{2.157}$$

where w_i states the importance weight of each EKF filter, $\bar{\mathbf{x}}_{t,i}$ and $P_{t,i}$ are the mean vector and the covariance matrix of the ith EKF filter. The complexity of this algorithm is of order of $\mathcal{O}(M^{N_g})$ and it necessitates a heavy load of calculation for a practical case in which the expected maximum range of landmarks are more than 40 meters and the number of landmarks are more than five.

Generally, the EKF based methods have poor performance if landmarks are close to the robot since nonlinearities of measurement models become more severe and the linear approximation needed for the EKF filter could be poor.

Inverse Depth Parametrization

Civera et al. proposed a new perspective in [CDM08] by using special way of parametrization of a landmark uncertainty which is known as the inverse depth parametrization (IDP). Civera et al. discussed that the distribution of the inverse of a depth of a landmark with an unknown depth looks like a Gaussian distribution. To include such distribution in an EKF filter, they proposed to parametrize the position of a landmark using a vector including six parameters:

$$\mathbf{y}_j = [x_j^L \; y_j^L \; z_j^L \; \theta_j^L \; \phi_j^L \; \rho_j^L]^T \tag{2.158}$$

The position of a landmark based on this vector can be obtained as follows:

$$\mathbf{x}^L = \begin{bmatrix} x_j^L \\ y_j^L \\ z_j^L \end{bmatrix} + \frac{1}{\rho_j^L}\mathbf{m}(\theta_j^L, \phi_j^L) \tag{2.159}$$

where:

$$\mathbf{m}(\theta_j^L, \phi_j^L) = \begin{bmatrix} \cos\theta_j^L \cos\phi_j^L \\ \cos\theta_j^L \sin\phi_j^L \\ \sin\theta_j^L \end{bmatrix} \tag{2.160}$$

They also used a quaternion to encode the orientation of the camera. The motion model they used is as follows:

$$\begin{aligned} \mathbf{x}_{k+1}^W &= \mathbf{x}_k^W + \mathbf{v}_k^W \\ \mathbf{q}_{k+1}^W &= \mathbf{q}(w_k^C) \times \mathbf{q}_k^{WC} \\ \mathbf{v}_{k+1}^W &= \mathbf{v}_k^W + \mathbf{a}^w \Delta t \\ \omega_{k+1}^C &= \omega_k^C + \alpha^C \Delta t \end{aligned} \tag{2.161}$$

where $\mathbf{x}_{k+1}^W = [x_{k+1}^W \; y_{k+1}^W \; z_{k+1}^W]$ is the position of camera in the world frame, $\mathbf{v}^W$ is the linear velocity of the camera with respect to the world frame and w^C is the angular velocity with respect to the camera frame. $\mathbf{a}^W$ and α^C are the

linear and angular accelerations of the camera with respect to the world and the camera frames. The acceleration parameters are assumed to be zero mean Gaussian processes.

To form the state space model, it is required to obtain the measurement model based on the state variables. The measurements are the projections of landmark positions on the image plane of a calibrated camera. As known, the position of a point such as $\mathbf{p} = [p_x\, p_y\, p_z]^T$ from the world coordinate in the camera coordinate will be as follows:

$$\mathbf{p}' = R^T(\mathbf{p} - \mathbf{x}^W) \tag{2.162}$$

The above equation can be rewritten using Eq. (2.159) as follows:

$$\mathbf{p}' = R^T \left[\rho^{L,i} \left(\begin{bmatrix} x^{L,i} \\ y^{L,i} \\ z^{L,i} \end{bmatrix} - \mathbf{x}^W \right) + \mathbf{m}(\theta^{L,i}, \phi^{L,i}) \right] \tag{2.163}$$

Consequently, the projection of $\mathbf{p}' = [p'_x\, p'_y\, p'_z]^T$ on the image plane of the camera will be as follows:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{p'_x}{p'_f} \\ \frac{p'_y}{p'_z} \end{bmatrix} \tag{2.164}$$

Based on the mentioned equations as soon as a new landmark is observed for the first time, six parameters concerning the landmark position are added into the state vector. All parameters except the inverse depth parameter can be calculated based on the camera pose and the measurement model. The block related to the uncertainties of the parameters in the covariance matrix can also be initialized using the linear approximation of the motion and measurement models to transform the uncertainty of the camera pose to the landmark uncertainties. The only parameter which cannot be initialized based on the camera pose and measurements is the inverse depth parameter. To cover a large range of uncertainties Civera et al. proposed an initial value $\rho^{L,i} = 0.1$ and a standard deviation $\sigma_{\rho^{L,i}} = 0.5$.

As we will discuss in the next chapters, the inverse depth parametrization method diverges in two cases: first, if the landmarks are located close to the camera (less

than 3 m) at their initialization time or the landmarks are observed at the low parallax angles which could result in the localization of the landmarks behind the camera or in another word negative depths.

Logarithmic Depth Parametrization Method

Parsely and Julier attempted to solve the problem of negative depths by proposing a logarithmic depth parametrization [PJ08]. In their parametrization method, they used a logarithmic depth parameter (LDP) such as l as follows:

$$l = -\ln d \tag{2.165}$$

It can simply be verified that the depth parameter cannot be negative under any condition. This kind of parametrization, however, gives rise to the problem that l cannot be modeled by a Gaussian distribution any more. To address this problem, they used second order EKF filters in which the second order terms of the Taylor expansion are used. Due to the high nonlinearity of the logarithm function near to the origin, the LDP method has a very poor performance to deal with the close landmarks. Additionally, the second order Kalman filter could diverge in the case of relatively high odometry noises.

Iterative Inverse Depth Parametrization Method

To address the problems of the IDP method, Tully et al. proposed to perform the update part of the EKF filter iteratively [TMKC08](IIDP). Later, we will show that at least five iterations are required to make IDP robust in the presence of close landmarks, which means the computation load increases five times more. Additionally, the problem of negative depth is not solved in this method.

Parallax Angle Parametrization

Zhao et al. in [ZHYD11] tried to tackle the negative depth problem of the inverse depth parametrization method by introducing a new parameterization method. They proposed a delayed method in the context of the optimization

based method (known in the computer vision society as bundle adjustment). They parametrized a landmark with two vectors from which the landmark is observed and the parallax angle between the vectors. In this method, a landmark position is presented as follows:

$$L_j : (\mathbf{x}_1^R, \mathbf{x}_2^R, \alpha_j) \tag{2.166}$$

where $\mathbf{x}_1^R$ and $\mathbf{x}_2^R$ are the robot pose instances and α_j is the parallax angle.

As soon as a landmark is initialized, it can be used to modify robot poses using a bundle adjustment (cost optimization) method. In this regard, it is required to predict the next measurements based on the mentioned parameterization. It means, given the robot pose $\mathbf{x}_3^R$ and the parallax parametrization, we are interested to predict the projection of the L_j on the camera screen: $(x_{j,3}, y_{j,3})$.

This method depends on the visual odometry having a high quality and cannot use the landmarks which are observed at low parallax angles as the whole formulation depends on the parallax angle.

Other Monocular SLAM Methods

In addition to the monocular SLAM methods already discussed in this section, there are many other methods which are based on delayed initialization of landmarks, e.g. [KM07], [SCG13], [MAT14] and [SC14]. These methods apply bundle adjustment (cost optimization) and mainly focus on the improvements of feature tracking modules and leveraging different software architectures letting the methods run in real-time through parallel processing. Generally, the delayed methods can only handle landmarks observed at high parallax angles. Consequently, they typically have poor performances in environments where there are not enough close landmarks such that they can be observed at high parallax angles.

2.3.5 Overall View of Different Fusion Methods

In table 2.1, different fusion methods discussed in this section are listed. The methods are compared based on the used estimation techniques, the types of mea-

surements, their complexity, the way they initialize newly observed landmarks and their robustness against odometry and measurement noise.

Table 2.1: Overall view of different fusion methods. $\sqrt{SAM}$: Square root smoothing and mapping, DI: Delayed landmark initialization, GSF: Gaussian sum filter, IDP: Inverse depth parameterization, IIDP: Iterative inverse depth parameterization, LDP: Logarithmic depth parameterization and PAP: Parallax angle parameterization.
Estimation: the estimation method. Measurement: types of measurements used in the methods. Complexity: Escalation of complexity of methods with respect to the number of landmarks. Initialization: initialization of landmarks. Robustness: Robustness of the methods against odometry and measurement noise. (-): poor, (+): good, (++): very good.

Method	Estimation	Measurement	Complexity	Initialization	Robustness
EKF-SLAM	EKF	Bearing+Range	Quadratical	Un-delayed	+
FastSLAM	PF+EKF	Bearing+Range	Linear	Un-delayed	+
GraphSLAM	Optimization	Bearing+Range	Quadratical	Un-delayed	++
$\sqrt{SAM}$	Optimization	Bearing+Range	Linear	Un-delayed	++
DI	EKF	Bearing	Quadratical	Delayed	-
GSF	EKF	Bearing	Exponential	Un-delayed	++
IDP	EKF	Bearing	Quadratical	Un-delayed	-
IIDP	EKF	Bearing	Quadratical	Un-delayed	+
LDP	EKF	Bearing	Quadratical	Un-delayed	+
PAP	Optimization	Bearing	Quadratical	Delayed	-

3 Visual Odometry

In chapter 2, some of the well-known feature matching and tracking techniques were discussed. In literature, these methods have mostly been investigated for feature matching between two images and not a sequence of images [MM12a] [ope11]. Additionally, for evaluation, typically, the images captured from planar surfaces have been used as in this case, ground truths can be obtained using homography transforms [HZ04]. These types of images are no proper test beds for the evaluation of the feature matching methods for visual odometry purposes. Since in visual odometry cases, we deal usually with the scenes which have high depth variations and different change of scale of objects when cameras move. On the contrary, in this chapter, we show how these methods can be used to track a set of features within a sequence of images. Furthermore, we evaluate the proposed trackers based on KITTI dataset [KIT15]. Next in this chapter, relative camera motion estimation based on N-point methods are investigated and new methods are proposed to enhance the performance of the methods against measurement noise. To this end, we firstly show why the eight point method has a poor performance in the presence of measurement noise and then we propose modifications to enhance its performance. Additionally, we derive a regularization term based on the second order statistics of essential matrix elements to enhance camera motion estimation significantly. This chapter is based on the two of our publications [MM13b] and [MM14].

3.1 Feature Tracking

In this section, feature tracking within a sequence of images is discussed. We investigate feature tracking based on the methods introduced in section 2.1 and compare them using four measures: (i) endurance, (ii) precision, (iii) recall and

(iv) measurement noise. The definitions of the measures will be presented soon. In subsection 3.1.1, the feature tracking method based on the feature matching techniques (SIFT, SURF, ORB and BRISK) is discussed. In subsection 3.1.2, the feature tracking method using the Lucas-Kande method is presented.

3.1.1 Feature Tracking Using Feature Matching Techniques

In feature matching methods, features and their descriptors are extracted from two images and the matches are found based on the descriptors. To implement feature tracking using SIFT, SURF, ORB and BRISK, the following procedure should be considered. We define a class c including a point $\mathbf{p}$, its descriptor vector $\mathbf{d}$ and its identity number id and denote the class as $c = \{\mathbf{p}, \mathbf{d}, id\}$. By defining this class, given a sequence of images such as $L_0, ..., L_N$, we can extract feature points $\{\mathbf{p}_i\}$ and their descriptors $\{\mathbf{d}_i\}$ from each image L_t and form the class instances $c_{t,i}$ such that $c_{t,i}.\mathbf{p} = \mathbf{p}_{t,i}$, $c_{t,i}.\mathbf{d} = \mathbf{d}_{t,i}$ and $c_{t,i}.id = i$, where $i = 1, ..., M_t$ and M_t is the number of extracted features from L_t. With these assumptions, the feature tracking procedure can be implemented as follows:

1. $t = 0$.
2. Extract $C_t = \{c_{t,1}, ..., c_{t,M_t}\}$.
3. Extract $C_{t+1} = \{c_{t+1,1}, ..., c_{t+1,M_{t+1}}\}$.
4. Create empty sets $\hat{C}_t$ and $\hat{C}_{t+1}$.
5. Fill $\hat{C}_t$ and $\hat{C}_{t+1}$ with the matched points as follows:
   $\forall c_{t,i} \in C_t$ find two matched features in C_{t+1} such as $c_{t+1,f}$ and $c_{t+1,s}$ with the minimum Euclidean distances d_f and d_s.
   If $d_f/d_s < \delta_m$ ($\delta_m < 1$ is a threshold parameter), $c_{t,i} \rightarrow \hat{C}_t$ and $c_{t+1,f} \rightarrow \hat{C}_{t+1}$.
6. $\forall c_{t+1,i} \in \hat{C}_{t+1}$: $c_{t+1,i}.id = c_{t,i}.id$ and $c_{t+1,i}.\mathbf{d} = c_{t,i}.\mathbf{d}$.
7. $C_{t+1} = \hat{C}_{t+1}$.
8. $t = t + 1$.
9. Go to 3.

In the step 5 of the algorithm, a matching method known as 2-nn (2 nearest neighbors) matcher is used. In this type of matcher no minimum threshold over the minimum euclidean distances are forced, but the measure to determine if there is a good match for a point, is taking into account how much the first and second matches are similar. In other words if the the distances of descriptors of the first and second matched candidates to the descriptor of a query point are almost similar, obviously, the risk of wrong matching substantially increases. Another important step is the step 6, in which the identity of features from the frame 0 are transformed to the tracked features to keep the track of features. Additionally, the descriptors of features become updated at each step with the newest descriptor of each feature. Updating descriptors makes it possible to track features in longer sequences.

3.1.2 Feature Tracking Based on Lucas-Kanade Method

Unlike feature matching methods, in the Lucas-Kande (LK) method, extracted corner features from one frame are tracked based on brightness constancy of the features and their surrounding points. Applying the optical flow method, however, gives rise typically to a high amount of outliers. To deal with this problem, we used forward and backward calculations of optical flows. In this method, given two consecutive frames, corner features from the first frame are tracked in the second frame. The corresponding points in the second frame are fed again to the LK tracker to find their correspondent points in the first frame. By applying the forward and backward Lucas-Kanade (FBLK), we expect to obtain the same feature positions in the first frame. This expectation can be used as a measure to reject most of the outliers.

By defining a class $c = \{\mathbf{p}, id, s\}$ where $\mathbf{p}$ is a feature point, id is its id and s represents its status (it could be inlier or outlier), the feature tracking procedure will be as follows:

1. $t = 0$.
2. Extract corner features from L_t and form the class instances $c_{t,id,s}$, where $id = 1...M_t$ and $s =$ inlier.

3. Track the features $c_{t,i}$ in the frame L_{t+1} and store them in $c_{t+1,i}$.
4. Track $c_{t+1,i}$ from the frame L_{t+1} in L_t and store them in $c'_{t,i}$.
5. Set the status of i^{th} feature as outlier if $||c_{t,i}.\mathbf{p} - c'_{t,i}.\mathbf{p}|| > \delta_p$.
6. $t = t + 1$.
7. Go to 3.

3.1.3 Comparison of Methods

In this section, we investigate the qualities of the feature tracking methods discussed in the previous section. The following measures for the evaluation of the methods are considered:

1. **Endurance**: Given N features extracted from frame 0, endurance is the ratio of the number of points tracked in the next frames to N.
2. **Precision**: The ratio of the number of correctly tracked features to the number of tracked features.
3. **Recall**: The ratio of the number of correctly tracked features to the number of all features which were supposed to be tracked.
4. **Measurement Noise**: The average distances of tracked features from their correspondent epipolar lines.

To evaluate the discussed methods, image sequences with the ground truth of camera poses are required. Thanks to the KITTI dataset [KIT15] provided for visual odometry, we were able to run such tests for a variety of scenes. In this dataset eleven long sequences with ground truths are given. The ground truth for each camera pose is formed in a 4×4 matrix as follows:

$$Pose_t = \begin{bmatrix} R_t & \mathbf{c}_t \\ 0 & 1 \end{bmatrix} \tag{3.1}$$

where R_t is a rotation matrix encoding the orientation of the camera at time t with respect to the first camera pose, and $\mathbf{c}_t = [c_x,\ c_y,\ c_z]^T$ shows the camera position in the coordinate frame attached to the first camera pose.

Since the depths of points are not given, the measure that we used to find how well a feature was correctly tracked was the distance of its matched point to its correspondent epipolar line. In case of uncalibrated cameras, we know that:

$$[x_t^u\ y_t^u\ 1]F\begin{bmatrix} x_0^u \\ y_0^u \\ 1 \end{bmatrix} = 0 \tag{3.2}$$

Given a point in the first frame $[x_{i,0}^u\ y_{i,0}^u]^T$ and the fundamental matrix F, the correspondent point $[x_{i,t}^u\ y_{i,t}^u]^T$ should lie on a line with the coefficients $[a\,b\,c]^T = F[x_{i,0}^u\ y_{i,0}^u\ 1]^T$. Therefore, the distance of $[x_{i,t}^u\ y_{i,t}^u]^T$ to its epipolar line will be a good measure to determine the quality of a tracker. The error is calculated as follows:

$$\epsilon_{i,t}^d = \frac{|ax_{i,t}^u + by_{i,0}^u + c|}{\sqrt{a^2+b^2}} \tag{3.3}$$

Based on the epipolar distances, a matching was considered correct if $\epsilon_t^d \leq 1$ and wrong if $\epsilon_t^d > 1$. As a result, the two following measures can be calculated:

$$\text{Precision}_t = \frac{(\text{Number of correct matches})_t}{(\text{Number of tracked features})_t} \tag{3.4}$$

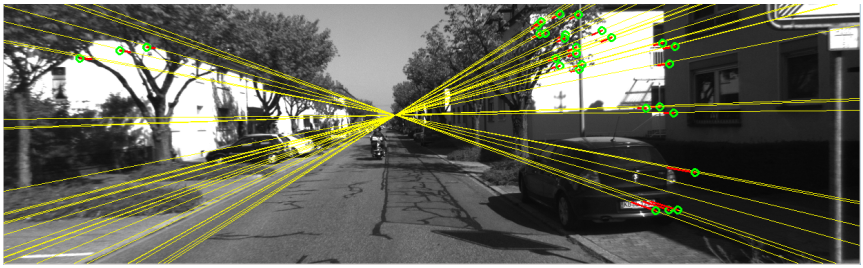
$$\begin{aligned} \text{Recall}_t &= \frac{(\text{Number of correct matches})_t}{(\text{Number of all features which could be tracked})_{t-1}} \\ &\approx \frac{(\text{Number of correct matches})_t}{(\text{Number of all features})_{t-1}} \end{aligned} \tag{3.5}$$

"Precision" reflects the the ability of a tracker to track features correctly, while "Recall" signifies the ability of a tracker not to lose the track of a features. In Eq. (3.5), an approximation for recall is used, where all features at time $t-1$ is taken into account instead of all features which could be tracked. The difference is that some features at time $t-1$ might be occluded at time t and there is no way to track them. Unfortunately, detection of occluded features based on epipolar geometry is not feasible. Nevertheless, as the number of occluded features are

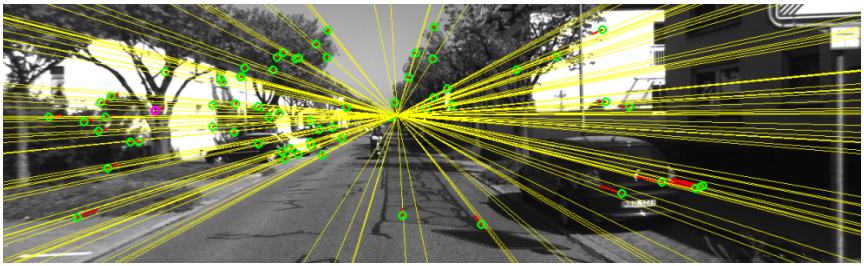
usually low in comparison to the non-occluded features, we can use the mentioned approximation and obtain a fair measure for the comparison of all methods. Another important measure for evaluation of a tracker is the average of the distances of matched points to their correspondent epipolar lines at each frame. The measure signifies measurement noise which affects adversely the quality of estimation of an essential or a fundamental matrix.

$$\text{Measurement Noise}_t = \frac{1}{N_t} \sum_{i=1}^{N_t} \epsilon_{i,t}^d \tag{3.6}$$

In Fig. 3.1 and Fig. 3.1, tracked features from the sequence 0 of the KITTI dataset can be seem. The epipolar lines are drawn with yellow lines, the inliers are shown with green circles and outliers with pink circles.

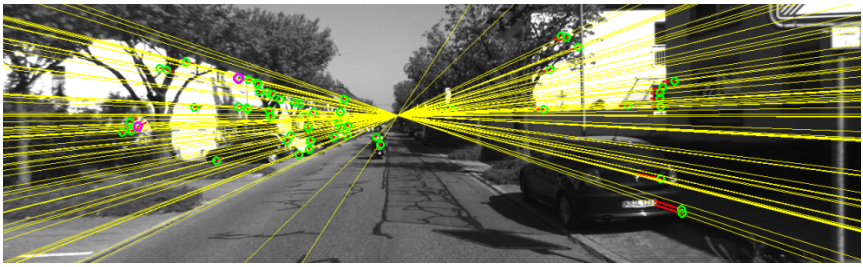


(a)

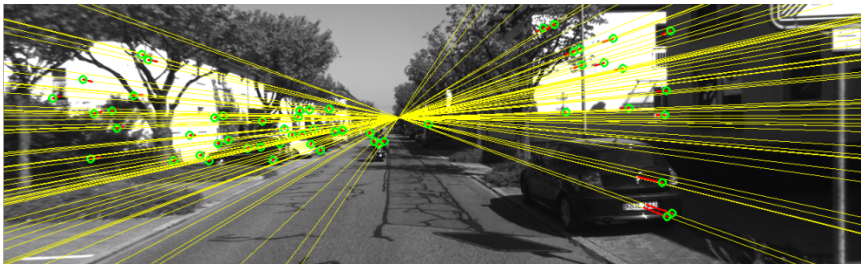


(b)

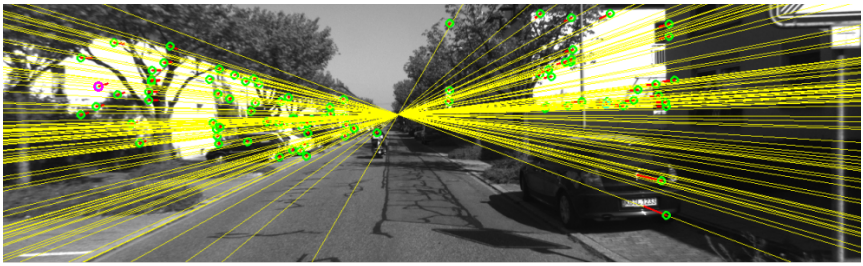
Figure 3.1: Feature tracking using (a) FBLK, (b) BRISK. The green circles are inliers and pink circles are outliers.



(a)



(b)



(c)

Figure 3.2: Feature tracking using and (a) ORB, (b) SIFT and (b) SURF. The green circles are inliers and pink circles are outliers.

In Fig. 3.3, the endurance measure of different tracking methods can be seen. We can see that the best endurance measure belongs to FBLK. SIFT, SURF, ORB and BRISK have the next positions. Surprisingly, we can see most of the features from the frame 0 cannot be tracked in frame one using ORB and BRISK. As a result, it can be concluded that the trackers based on ORB and BRISK are not appropriate for the multiple frame optimizations needed in SLAM algorithms.

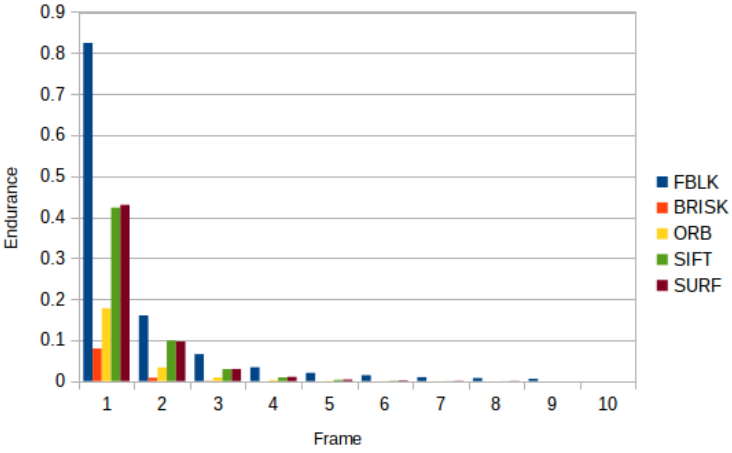


Figure 3.3: Endurance measure of different tracking methods.

In Fig. 3.4, the precision and recall measures can be seen. FBLK outperforms the other methods both in terms of precision and recall. In average, SIFT, ORB, SURF and BRISK can be ranked respectively in term of precision. Concerning the recall measure, however, the ranking is as follows: FBLK, SURF, SIFT, ORB and BRISK. We see that all methods except FBLK fail to track many features from the frame zero in frame one. This is due to the fact that the scale of close features change noticeably between the first two frames and in this case many of them cannot be detected as features in the second frame due to the discrete sampling of scale-space.

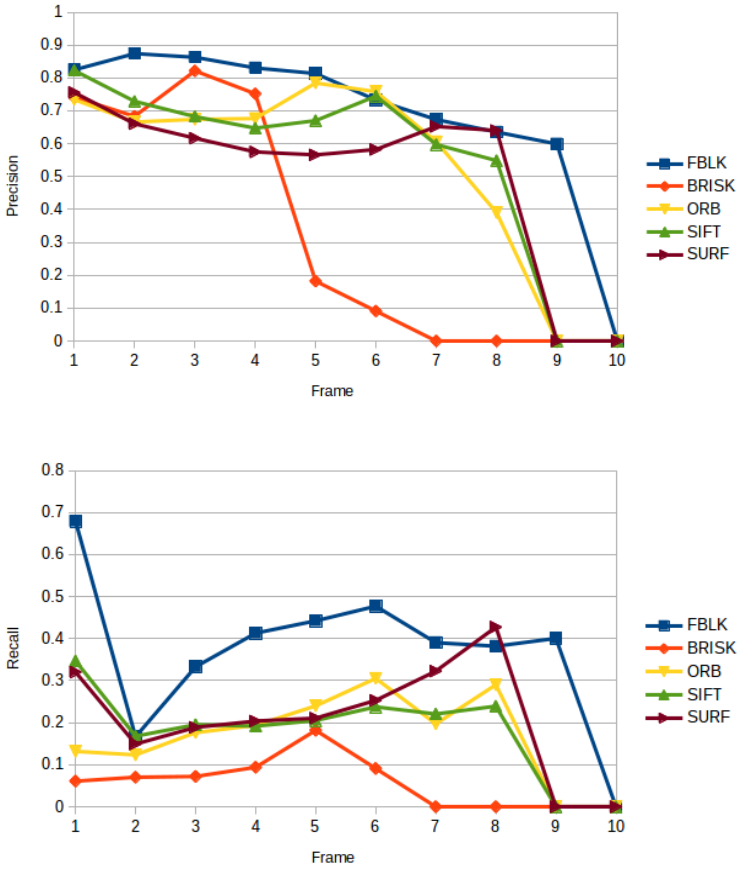


Figure 3.4: Precision and Recall measures for different feature tracking methods.

Among the matching based methods again ORB and Brisk have the poorest performances as their descriptors are not as descriptive of SIFT and SURF, resulting in rejection of many matches. Additionally, we see that FBLK is not also much robust against the changes of scales and in the second frame cannot track many of the features of which scales have changed significantly.

The comparison of the methods in terms of measurement noise is depicted in Fig. 3.5. We can see that until frame 2, FBLK has better performance, but for next frames, BRISK, ORB, SIFT and SURF have better performances. It lies in the fact that the cornerness (based on Harris corner measure [HS88]) of the tracked points are not verified explicitly at each time step when using FBLK. As a result, the tracked points may drift from their real positions gradually. Conversely, feature matching techniques localize features at each step which avoid drifts in localization of features. As ORB and BRISK use corner features, we can see that they have better performance than SIFT and SURF which use blob features. These results are reasonable since blob centers cannot be calculated as accurately as corner feature locations.

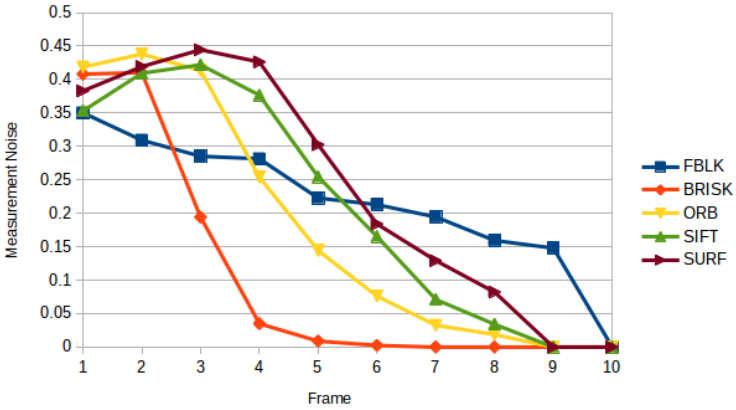


Figure 3.5: Measurement noise of different tracking methods.

Table 3.1: Elapsed time for feature matching between two frames for an average feature numbers equal to 1500.

Tracker	FBLK	BRISK	ORB	SIFT	SURF
E. T. [ms]	10	248.3	96.3	493.6	284.3

The elapsed time of each tracker for tracking 1000 features between two frames can be seen. We see that the FBLK tracker also outperforms the other methods to a great extent.

The average of the four different measures are shown in table 3.2. Obviously, the FBLK outperforms the other methods in term of endurance, precision and recall. The measurement noise of FBLK is only a bit worse than that of the BRISK method. As a result, we selected the FBLK feature tracker for the visual odometry and visual SLAM methods, which will be proposed in this and the next chapter.

Table 3.2: The average of the measures (endurance, precision, recall and measurement noise) for different trackers.

Method	FBLK	BRISK	ORB	SIFT	SURF
Endurance	**0.37**	0.06	0.16	0.18	0.20
Precision	**0.68**	0.33	0.52	0.54	0.46
Recall	**0.37**	0.06	0.16	0.18	0.20
Measurement Noise	0.21	**0.18**	0.22	0.26	0.30

3.2 Inconsistencies of N-point Methods for Camera Motion Estimation

A practical N-point ($N = 8, 7$ or 5) method in the context of a RANSAC algorithm [FB81] consists of three steps.

1. Estimation of several essential matrices based on the random selection of the minimum required number of matching points (minimal set).

2. Checking the validity of the estimated essential matrices and selecting the best essential matrix which minimizes coplanarity constraints for all matching points.

3. Extraction of the rotation matrix and translation vector using the SVD decomposition and selection of the feasible solution (from four algebraic solutions) using the Cheirality test [HZ04]. This test determines which solution back-projects matched points in front of both cameras.

In the following, we discuss that all the methods have a shortcoming in the first and third steps and in the next section we address the shortcomings. In essential matrix estimation, one common problem among all N-point methods is that the deviation of coplanarity equations from zero in the presence of measurement noise has not been considered. Given the correct essential matrix, in the presence of measurement noise coplanarity equations may not be zero. However, we can calculate how much each equation can deviate from zero and solve the obtained homogeneous equation system in the sense of Mahalanobis distances based on the presumed deviations.

As we already discussed, using eight matched points, we can also estimate the essential matrix by forcing $e_9 = 1$, which works generally better than the SVD based technique. However, it can be shown that if the camera has a constrained motion of wheeled vehicles, e_9 will be zero. Under this constraint, the translation vector has only the forward translation at each step: $\mathbf{t} = [0, 0, t_z]^T$. Given this translation vector we have:

$$E = RT = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \begin{bmatrix} 0 & t_z & 0 \\ -t_z & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -r_2 t_z & r_1 t_z & 0 \\ -r_5 t_z & r_4 t_z & 0 \\ -r_8 t_z & r_7 t_z & 0 \end{bmatrix} \quad (3.7)$$

It can be seen that the whole third column of E becomes zero and consequently, the coplanarity equation cannot be divided by e_9. In practice, however, t_x and t_y may take small values. In this case, the other nonzero elements of E should take very large values which may give rise to numerical errors. Therefore, we can expect that this type of the 8-point method will have much poor performance than the SVD based 8-point method. As already discussed in section 2.2.4, for the 7- and 5- point methods, due to forcing one unknown to 1, we may encounter the numerical errors, if the unknown should be zero or near to zero.

The best essential matrix can be obtained based on the distances of the matched points to their epipolar lines. The essential matrix which results in the minimum distances for all matched points is considered as the best matrix [HZ04].

After obtaining the best E matrix, we can obtain the rotation matrix and the translation vector using the SVD decomposition. This decomposition results in four mathematically valid solutions; however, only one of them can back project the image points in space in front of both camera poses. This measure is known as the Cheirality test. Unfortunately, this test fails if the space points are located relatively far from the camera and there exist measurement noise. In such cases, the points could be wrongly back projected behind the cameras. This situation occurs in outdoor scenarios very often.

3.3 Modified N-Point Methods

In this section, we address the shortcomings of different N-point methods mentioned in the previous section.

3.3.1 Considering Measurement Noise in the Estimation of E

The first step of different N-point methods is finding the null space vectors. In this regard, a homogeneous equation system including N equations is formed as follows:

$$A\mathbf{e} = \mathbf{0} \tag{3.8}$$

where A is a $N \times 9$ matrix of which rows contain coefficients of coplanarity equations and $\mathbf{e} = [e_1 \ e_2 \ ... \ e_9]^T$. The above equation system can be solved under the constraint $\mathbf{e}^T\mathbf{e} = 1$. Considering the SVD decomposition of A:

$$A = U\Sigma V^T \tag{3.9}$$

the column of V^T which is correspondent to the smallest singular value will be the solution of the equation system. In fact SVD approach minimizes the cost function $C = (A\mathbf{e})^T(A\mathbf{e})$, which is the square of euclidean distances between $A\mathbf{e}$ and the origin. On the other hand, we know if we deal with noisy data, using the Mahalanobis distance gives better results since the proper deviation of each coplanarity equation from zero will intuitively be taken into account. As a result, the following cost function should be minimized:

$$C = (A\mathbf{e})^T\Lambda(A\mathbf{e}) \tag{3.10}$$

where Λ is the inverse of the covariance related to the coplanarity equations.

$$\Lambda_{i,j}^{-1} = \begin{cases} var(\mathbf{v}_{i,2}^T E \mathbf{v}_{i,1}) & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases} \tag{3.11}$$

where i is the index of i^{th} matched points. Since E is not initially known it is not straightforward to calculate the diagonal elements. However, we can consider large uncertainties for the rotation and translation parameters with zero mean values. If we parametrize a rotation matrix with the angles $\phi \sim \mathcal{N}(0, \sigma_\phi^2)$, $\theta \sim \mathcal{N}(0, \sigma_\theta^2)$ and $\psi \sim \mathcal{N}(0, \sigma_\psi^2)$ we can write:

$$R = \begin{bmatrix} c\theta\, c\psi & -c\phi\, s\psi - s\phi\, s\theta\, c\psi & s\phi\, s\psi - c\phi\, s\theta\, c\psi \\ c\theta\, s\psi & c\phi\, c\psi - s\phi\, s\theta\, s\psi & -s\phi\, c\psi - c\phi\, s\theta\, s\psi \\ s\theta & s\phi\, c\theta & c\phi\, c\theta \end{bmatrix} \tag{3.12}$$

where c and s stand for cos and sin consecutively. Since in practical cases the amount of these angles are less than $\frac{\pi}{4}$ rad, we can use the following approximation of R based on the Taylor expansion and ignoring terms with the orders more than two:

$$R \approx \begin{bmatrix} 1 - \frac{\theta^2}{2} - \frac{\psi^2}{2} & -\psi - \phi\theta & -\theta + \phi\psi \\ \psi & 1 - \frac{\phi^2}{2} - \frac{\psi^2}{2} & -\phi - \theta\psi \\ \theta & \phi & 1 - \frac{\phi^2}{2} - \frac{\theta^2}{2} \end{bmatrix} \tag{3.13}$$

Since $E = RT$, we have:

$$\begin{aligned} e_1 &= -t_z(\psi + \phi\theta) + t_y(\theta - \phi\psi) \\ e_2 &= t_x(\phi\psi - \theta) + t_z(\frac{\psi^2 + \theta^2}{2} - 1) \\ e_3 &= t_y(1 - \frac{\psi^2 + \theta^2}{2}) + t_x(\psi + \phi\theta) \\ e_4 &= -t_z(\frac{\phi^2 + \psi^2}{2} - 1) + t_y(\phi + \psi\theta) \\ e_5 &= -t_z\psi - t_x(\phi + \psi\theta) \\ e_6 &= \psi t_y + t_x(\frac{\phi^2 + \psi^2}{2} - 1) \\ e_7 &= t_z\phi + t_y(\frac{\phi^2 + \theta^2}{2} - 1) \\ e_8 &= -\theta t_z - t_x(\frac{\phi^2 + \theta^2}{2} - 1) \\ e_9 &= -\phi t_x + \theta t_y \end{aligned} \tag{3.14}$$

Based on the assumption that the absolute values of rotation parameters are less than $\frac{\pi}{4}$ rad, $\sigma_\phi = \sigma_\theta = \sigma_\psi = 0.5$ are reasonable standard deviation values for the parameters. Regarding the uncertainty of the translation vector, it can be

verified if the equation system is solved using the SVD method, the following constraint holds:

$$t_x^2 + t_y^2 + t_z^2 = \frac{1}{2} \tag{3.15}$$

It means that $[t_x, t_y, t_z]^T$ has a uniform distribution over a sphere with the radius $\frac{1}{\sqrt{2}}$:

$$p(t_x, t_y, t_z) = \begin{cases} \frac{1}{2\pi} & \text{if } t_x^2 + t_y^2 + t_z^2 = \frac{1}{2} \\ 0 & \text{else} \end{cases} \tag{3.16}$$

To calculate the mean values and variances of translation elements, we may need to marginalize out two variables from the above distribution. However, due to the symmetry of the distribution, it can be inferred that:

$$\begin{aligned} \mu_{t_x} &= \mathcal{E}(t_x) = 0 \\ \mu_{t_y} &= \mathcal{E}(t_y) = 0 \\ \mu_{t_z} &= \mathcal{E}(t_z) = 0 \end{aligned}$$

where $\mathcal{E}(.)$ is the expectation operator.

On the other hand, by applying expectation on Eq. (3.15), we will have:

$$\mathcal{E}(t_x^2) + \mathcal{E}(t_y^2) + \mathcal{E}(t_z^2) = \frac{1}{2} \tag{3.17}$$

Since the mean values of the translation elements are zero, we have:

$$\sigma_{t_x}^2 + \sigma_{t_y}^2 + \sigma_{t_z}^2 = \frac{1}{2} \tag{3.18}$$

Due to the symmetry of the distribution, we can also say: $\sigma_{t_x} = \sigma_{t_y} = \sigma_{t_z}$. As a result:

$$\sigma_{t_x}^2 = \sigma_{t_y}^2 = \sigma_{t_z}^2 = \frac{1}{6} \tag{3.19}$$

If we consider measurement noises for the image coordinates we have:

$$\begin{aligned} x_i &= \bar{x}_i + \tilde{x}_i \\ y_i &= \bar{y}_i + \tilde{y}_i; \quad i = 1, 2 \end{aligned} \tag{3.20}$$

where (x_i, y_i) and $(\bar{x}_i, \bar{y}_i)$ are the measured and real feature positions respectively. $\tilde{x}_i$ and $\tilde{y}_i$ are zero mean normal random variables modeling measurement noise. Based on the mentioned definitions, now we are interested to find the following variance:

$$\begin{aligned} \sigma_c^2 =& var(\mathbf{v}_2^T E \mathbf{v}_1) \\ \approx& var([\bar{x}_2\ \bar{y}_2\ 1]E[\bar{x}_1\ \bar{y}_1\ 1]^T) \\ &+ var\left([\bar{x}_2\ \bar{y}_2\ 1]E[\tilde{x}_1\ \tilde{y}_1\ 0]^T\right) + var\left([\tilde{x}_2\ \tilde{y}_2\ 0]E[\bar{x}_1\ \bar{y}_1\ 1]^T\right) \end{aligned} \tag{3.21}$$

Eq. (3.21) has three terms, but the last two terms are dependent on the measurement noise. These terms provide reasonable measures presenting how much each coplanarity equation can deviate from zero. It means that we are interested to force the first term to zero. In [MM13b], we used the last two terms to obtain the deviation measure. Furthermore, since all the random variables are less than one, the variances of higher order terms vanish quickly comparing to the lower order terms. Therefore, using Eq. (3.7) and Eq. (3.13) and by ignoring the terms with the orders more than two we have:

$$\sigma_c^2 \approx var\left(\begin{bmatrix} -t_z(\phi + y_2) + t_y \\ t_z(\theta + x_2) - t_x \end{bmatrix}^T \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \end{bmatrix}\right) + var\left(\begin{bmatrix} -\psi t_x - t_y + y_1 t_z \\ t_x - \psi t_y - x_1 t_z \end{bmatrix}^T \begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \end{bmatrix}\right) \tag{3.22}$$

It should be noted that $(\bar{x}_i, \bar{y}_i)$ in Eq. (3.21) is replaced with (x_i, y_i) in Eq. (3.22) as only noisy measurements are available to us. The effect of these replacements, however, can simply be overlooked as the the noise terms are generally very small, and they vanish quickly by being multiplied with other noise terms. Applying

the variance operation and taking into account that $\sigma_{t_x}^2 = \sigma_{t_y}^2 = \sigma_{t_z}^2 = \frac{1}{6}$, we will have:

$$\sigma_c^2 = \frac{\sigma_p^2}{6}\big(4 + \sigma_\phi^2 + \sigma_\theta^2 + 2\sigma_\psi^2 + (1 + \sigma_\phi^2 + \sigma_\theta^2 + \sigma_\psi^2)(x_1^2 + y_1^2 + x_2^2 + y_2^2)\big) \quad (3.23)$$

Since σ_p^2 is a common constant for all equations, it can be removed and then each equation should be divided by its relevant σ_c to solve the equation system in Eq. (3.8) based on the Mahalanobis distance.

3.3.2 Including Regularization Term

In the previous section, we saw that we can consider limited uncertainties for the motion parameters based on the physical constraints of camera motions. As a result, we can also calculate uncertainties for essential matrix elements in the form of a covariance matrix such as $P_{\mathbf{e}}$ for the elements. The covariance matrix can be used to regularize the solution of equation system for different N-point methods. This method was published in [MM14].

Considering the approximated essential matrix elements in Eq. (3.14), the mean values and variances of rotation and translation parameters, we can calculate first and second order statistics of essential matrix elements:

$$\mu_{\mathbf{e}} = \mathcal{E}(\mathbf{e}) = \mathbf{0} \quad (3.24)$$

and

$$P_{\mathbf{e}} = \mathcal{E}\left((\mathbf{e} - \mu_{\mathbf{e}})(\mathbf{e} - \mu_{\mathbf{e}})^T\right) = \mathcal{E}(\mathbf{e}\mathbf{e}^T) \quad (3.25)$$

By plugging Eq. (3.14) in Eq. (3.25), we obtain the following covariance matrix:

$$P_{\mathbf{e}} = \begin{bmatrix} \sigma_{e_1}^2 & 0 & 0 & 0 & \sigma_{e_1,e_5} & 0 & 0 & 0 & \sigma_{e_1,e_9} \\ 0 & \sigma_{e_2}^2 & 0 & \sigma_{e_2,e_4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{e_3}^2 & 0 & 0 & 0 & \sigma_{e_3,e_7} & 0 & 0 \\ 0 & \sigma_{e_2,e_4} & 0 & \sigma_{e_4}^2 & 0 & 0 & 0 & 0 & 0 \\ \sigma_{e_1,e_5} & 0 & 0 & 0 & \sigma_{e_5}^2 & 0 & 0 & 0 & \sigma_{e_5,e_9} \\ 0 & 0 & 0 & 0 & 0 & \sigma_{e_6}^2 & 0 & \sigma_{e_6,e_8} & 0 \\ 0 & 0 & \sigma_{e_3e_7} & 0 & 0 & 0 & \sigma_{e_7}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{e_6e_8} & 0 & \sigma_{e_8}^2 & 0 \\ \sigma_{e_1e_9} & 0 & 0 & 0 & \sigma_{e_5,e_9} & 0 & 0 & 0 & \sigma_{e_9}^2 \end{bmatrix}$$

where

$$\begin{aligned}
\sigma_{e_1}^2 =& \sigma_\phi^2\sigma_\psi^2\sigma_{t_y}^2 + \sigma_\phi^2\sigma_\theta^2\sigma_{t_z}^2 + \sigma_\psi^2\sigma_{t_z}^2 + \sigma_\theta^2\sigma_{t_y}^2, \\
\sigma_{e_1e_5} =& \sigma_\psi^2\sigma_{t_z}^2, \\
\sigma_{e_1e_9} =& \sigma_\theta^2\sigma_{t_y}^2, \\
\sigma_{e_2}^2 =& \sigma_\phi^2\sigma_\psi^2\sigma_{t_x}^2 + \frac{3}{4}\sigma_\psi^4\sigma_{t_z}^2 + \frac{1}{2}\sigma_\psi^2\sigma_\theta^2\sigma_{t_z}^2 - \sigma_\psi^2\sigma_{t_z}^2 + \frac{3}{4}\sigma_\theta^4\sigma_{t_z}^2 + \sigma_\theta^2\sigma_{t_x}^2 \\
& - \sigma_\theta^2\sigma_{t_z}^2 + \sigma_{t_z}^2, \\
\sigma_{e_2e_4} =& -\frac{1}{4}\sigma_\phi^2\sigma_\psi^2\sigma_{t_z}^2 - \frac{1}{4}\sigma_\phi^2\sigma_\theta^2\sigma_{t_z}^2 + \frac{1}{2}\sigma_\phi^2\sigma_{t_z}^2 - \frac{3}{4}\sigma_\psi^4\sigma_{t_z}^2 - \frac{1}{4}\sigma_\psi^2\sigma_\theta^2\sigma_{t_z}^2 \\
& + \sigma_\psi^2\sigma_{t_z}^2 + \frac{1}{2}\sigma_\theta^2\sigma_{t_z}^2 - \sigma_{t_z}^2, \\
\sigma_{e_3}^2 =& \sigma_\phi^2\sigma_\theta^2\sigma_{t_x}^2 + \frac{3}{4}\sigma_\psi^4\sigma_{t_y}^2 + \frac{1}{2}\sigma_\psi^2\sigma_\theta^2\sigma_{t_y}^2 + \sigma_\psi^2\sigma_{t_x}^2 - \sigma_\psi^2\sigma_{t_y}^2 + \frac{3}{4}\sigma_\theta^4\sigma_{t_y}^2 \\
& - \sigma_\theta^2\sigma_{t_y}^2 + \sigma_{t_y}^2, \\
\sigma_{e_3e_7} =& -\frac{1}{4}\sigma_\phi^2\sigma_\psi^2\sigma_{t_y}^2 - \frac{1}{4}\sigma_\phi^2\sigma_\theta^2\sigma_{t_y}^2 + \frac{1}{2}\sigma_\phi^2\sigma_{t_y}^2 - \frac{1}{4}\sigma_\psi^2\sigma_\theta^2\sigma_{t_y}^2 + \frac{1}{2}\sigma_\psi^2\sigma_{t_y}^2 \\
& - \frac{3}{4}\sigma_\theta^4\sigma_{t_y}^2 + \sigma_\theta^2\sigma_{t_y}^2 - \sigma_{t_y}^2, \\
\sigma_{e_4}^2 =& \frac{3}{4}\sigma_\phi^4\sigma_{t_z}^2 + \frac{1}{2}\sigma_\phi^2\sigma_\psi^2\sigma_{t_z}^2 + \sigma_\phi^2\sigma_{t_y}^2 - \sigma_\phi^2\sigma_{t_z}^2 + \frac{3}{4}\sigma_\psi^4\sigma_{t_z}^2 + \sigma_\psi^2\sigma_\theta^2\sigma_{t_y}^2 \\
& - \sigma_\psi^2\sigma_{t_z}^2 + \sigma_{t_z}^2,
\end{aligned}$$

$$
\begin{aligned}
\sigma_{e_5}^2 =& \sigma_\phi^2\sigma_{t_x}^2 + \sigma_\psi^2\sigma_\theta^2\sigma_{t_x}^2 + \sigma_\psi^2\sigma_{t_z}^2,\\
\sigma_{e_5e_9} =& \sigma_\phi^2\sigma_{t_x}^2,\\
\sigma_{e_6}^2 =& \frac{3}{4}\sigma_\phi^4\sigma_{t_x}^2 + \frac{1}{2}\sigma_\phi^2\sigma_\psi^2\sigma_{t_x}^2 - \sigma_\phi^2\sigma_{t_x}^2 + \frac{3}{4}\sigma_\psi^4\sigma_{t_x}^2 - \sigma_\psi^2\sigma_{t_x}^2 + \sigma_\psi^2\sigma_{t_y}^2 + \sigma_{t_x}^2,\\
\sigma_{e_6,e_8} =& -\frac{3}{4}\sigma_\phi^4\sigma_{t_x}^2 - \frac{1}{4}\sigma_\phi^2\sigma_\psi^2\sigma_{t_x}^2 - \frac{1}{4}\sigma_\phi^2\sigma_\theta^2\sigma_{t_x}^2 + \sigma_\phi^2\sigma_{t_x}^2 - \frac{1}{4}\sigma_\psi^2\sigma_\theta^2\sigma_{t_x}^2\\
&+ \frac{1}{2}\sigma_\psi^2\sigma_{t_x}^2 + \frac{1}{2}\sigma_\theta^2\sigma_{t_x}^2 - \sigma_{t_x}^2,\\
\sigma_{e_7}^2 =& \frac{3}{4}\sigma_\phi^4\sigma_{t_y}^2 + \frac{1}{2}\sigma_\phi^2\sigma_\theta^2\sigma_{t_y}^2 - \sigma_\phi^2\sigma_{t_y}^2 + \sigma_\phi^2\sigma_{t_z}^2 + \frac{3}{4}\sigma_\theta^4\sigma_{t_y}^2 - \sigma_\theta^2\sigma_{t_y}^2 + \sigma_{t_y}^2,\\
\sigma_{e_8}^2 =& \frac{3}{4}\sigma_\phi^4\sigma_{t_x}^2 + \frac{1}{2}\sigma_\phi^2\sigma_\theta^2\sigma_{t_x}^2 - \sigma_\phi^2\sigma_{t_x}^2 + \frac{3}{4}\sigma_\theta^4\sigma_{t_x}^2 - \sigma_\theta^2\sigma_{t_x}^2 + \sigma_\theta^2\sigma_{t_z}^2 + \sigma_{t_x}^2 \text{and}\\
\sigma_{e_9}^2 =& \sigma_\phi^2\sigma_{t_x}^2 + \sigma_\theta^2\sigma_{t_y}^2.
\end{aligned}
$$

In Eq. 3.26, the dependencies of the essential matrix elements pop up as nonzero off-diagonal elements. According to the smoothing method proposed in [DK06], we can leverage a smoothing technique to estimate an essential matrix by minimizing an appropriate cost function. We define a cost function consisting of the coplanarity equations as data terms and the covariance matrix in Eq. 3.26 as a smoothness term:

$$C = \sum_{i=1}^{N} \frac{1}{\sigma_{c_i}^2}(A_i\mathbf{e})^T(A_i\mathbf{e}) + \mathbf{e}^T P_e^{-1}\mathbf{e} \tag{3.26}$$

where N is the number of matched points, A_i is a row vector of i^{th} coplanarity coefficients and σ_{c_i} is the standard deviation for the i^{th} coplanarity equation.

In Eq. (3.23), the calculation of σ_{c_i} is shown. It is necessary to mention that in Eq. (3.23), the inverse of σ_p determines the importance weight of the data term: the larger σ_p, the more estimations of essential matrix elements are affected by the regularization term. Therefore, σ_p should be chosen to attain a balance between the data and regularization terms such that the method can work well for a wide range of measurement noise. Based on simulation and experimental results, we found $\sigma_p = 0.5$ pixel is a proper selection.

In order to minimize the cost function in Eq. 3.26, we rewrite it as follows:

$$C = (B\mathbf{e})^T P^{-1} (B\mathbf{e}) \tag{3.27}$$

where

$$B = \begin{bmatrix} A_1 \\ \vdots \\ A_N \\ I_{9\times 9} \end{bmatrix} \quad \text{and} \quad P = \begin{bmatrix} \sigma_{c,1}^2 & 0 & 0 & 0 \\ \vdots & \ddots & ... & 0 \\ 0 & 0 & \sigma_{c,N}^2 & 0 \\ 0 & 0 & 0 & P_e \end{bmatrix} \tag{3.28}$$

Now we should find the vectors which minimize the cost function. To this end, we utilize the SVD decomposition for $P^{-1/2}B$:

$$P^{-1/2}B = U\Sigma V^T \tag{3.29}$$

Consequently, the columns of V corresponding to the smallest eigenvalues form the basis for different N-point methods.

3.3.3 Modified Cheirality Test

The last modification to the 8-point method concerns the Cheirality test. After the calculation of E, firstly, it should be decomposed as follows:

$$E = U\Sigma V^T \tag{3.30}$$

Consequently, the following solutions for R and $\mathbf{t}$ are obtained:

$$\begin{aligned} R_1 = UWV^T; \quad R_2 = UW^TV^T \\ T_1 = VZV^T; \quad T_2 = -VZV^T \end{aligned} \tag{3.31}$$

where

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \quad Z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.32}$$

Given the obtained rotation matrices and translation vectors, four valid solutions can be obtained: $\{R_1, \mathbf{t}_1\}$, $\{R_1, \mathbf{t}_2\}$, $\{R_2, \mathbf{t}_1\}$ and $\{R_2, \mathbf{t}_2\}$. However, the real solution should back project the image points in front of both camera positions. For matched points (x_1, y_1) and (x_2, y_2), we denote the position of the 3D point in space in the first and second camera frame as $\mathbf{p}_1 = [p_{1,x}\ p_{1,y}\ p_{1,z}]^T$ and $\mathbf{p}_2 = [p_{2,x}\ p_{2,y}\ p_{2,z}]^T$. We know that:

$$\begin{aligned} \mathbf{p}_1 &= d_1 \mathbf{v}_1 \\ x_2 &= \frac{\mathbf{r}_1(\mathbf{p}_1 - \mathbf{t})}{\mathbf{r}_3(\mathbf{p}_1 - \mathbf{t})} \end{aligned} \tag{3.33}$$

where d_1 is the depth of the point in the first camera frame, $\mathbf{v}_1 = [x_1\ y_1\ 1]^T$ and $\mathbf{r}_i$ is the i^{th} row of a rotation matrix. As a result, we will have:

$$d_1 = \frac{(\mathbf{r}_1 - x_2\mathbf{r}_3).\mathbf{t}}{(\mathbf{r}_1 - x_2\mathbf{r}_3).\mathbf{v}_1} \tag{3.34}$$

Similarly, the depth of the point in the second camera frame (d_2) is obtained. As a result, the solution which yields positive d_1 and d_2, should be the feasible solution. Nevertheless, generally, triangulation is very sensitive to measurement noise if the landmarks are relatively far from the camera or the translation amount is small. In these cases, the points which are in front of the cameras could be wrongly localized behind the camera.

To solve this problem, we consider the elements of the rotation matrix. Considering a rotation matrix which is parametrized based on rotation angles (Eq. (3.12)), it can be seen that if the rotation angles reside in the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$, r_1 and r_9 should be always greater than zero. Using this criterion, we can easily choose the correct rotation matrix. Then, for the selection of the translation vector we can use Eq. (3.34).

3.4 Simulation

In order to evaluate the proposed methods in the previous sections, we simulated a camera with the resolution 2000×2000 [pixels^2] and a focal length of 1000 pixels. The camera was moved based on random translation and rotation parameters

for two different types of motions: dominant forward ($t_z > t_x, t_y$) and dominant side translations ($t_x, t_y > t_z$). The rotation parameters ϕ, θ and ψ in both cases were generated randomly from a Gaussian distribution with the mean value 0.3 rad and standard deviation 0.1 rad. The camera could observe spatial points randomly distributed at depths from 10 to 20 meters. To simulate measurement noise, the projection of the spatial points on the camera screen was added by zero mean Gaussian noises with different standard deviation.

For the comparison, we have implemented the original N-point methods (named as 8-point-N, 7-point and 5-point) in MATLAB. 8-point-N stands for the normalized 8-point method proposed in [Har97]. We also implemented our proposed N-point versions: 8-point method (named as 8-point-M) in the subsection 3.3.1 and the 8-, 7- and 5- point methods based on the regularization term (named as 8-point-R, 7-point-R and 5-point-R) proposed in 3.3.2. The standard deviations of the rotation angles (σ_ϕ, σ_θ and σ_ψ) for the regularization constraint were set to 0.5 rad. For the evaluation, two different measures were used: the mean of magnitudes of errors between the estimated translation and true translation vectors ($MME_{\mathbf{t}}$) and the mean of magnitudes of errors between the estimated rotation angles and the true angles (MME_a). Assuming that the estimated translation vector at time k is $\hat{\mathbf{t}_k}$ and the ground truth is $\mathbf{t}_k$, the error between two vectors will be $\epsilon_{\mathbf{t},k} = \hat{\mathbf{t}_k} - \mathbf{t}_k$. Then, we have:

$$MME_{\mathbf{t}} = \frac{1}{K} \sum_{k=1}^{K} \sqrt{\epsilon_{\mathbf{t},k}^T \epsilon_{\mathbf{t},k}} \tag{3.35}$$

where K is the number of frames. If the estimated rotation matrix at time k is $\hat{R}_k$ and the ground truth is R_k the rotation matrix error will be:

$$R_{e,k} = R_k^T \hat{R}_k \tag{3.36}$$

Consequently, using Eq. 3.13, we can calculate the angle error as:

$$\epsilon_{a,k} = \frac{180}{\pi} \sqrt{3 - trace(R_{e,k})} \tag{3.37}$$

The mean of magnitudes of errors for angles will be:

$$MME_a = \frac{1}{K}\sum_{k=1}^{K}\epsilon_{a,k} \tag{3.38}$$

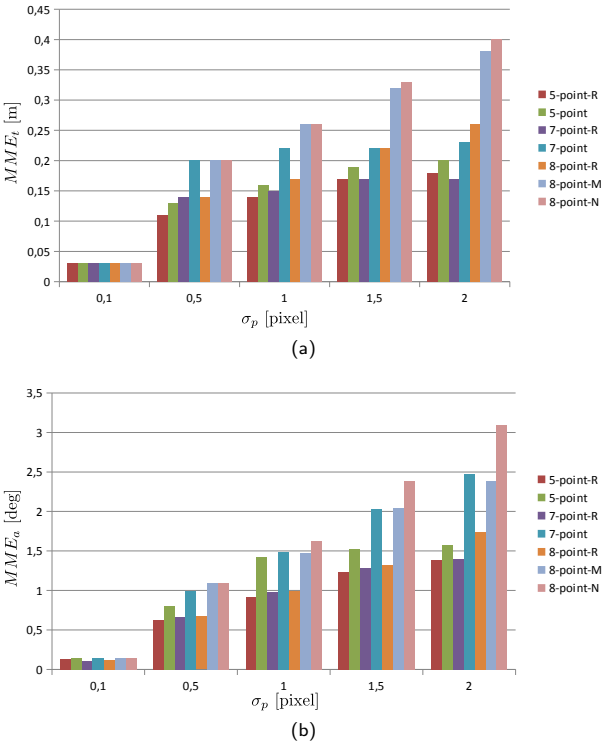


Figure 3.6: Mean of magnitudes of errors for translation in case of dominant forward translations.

It is good to mention that since the regularization term is centered on the origin; not surprisingly, the proposed method could perform well if the rotation parameters were also distributed about the origin. Therefore, we selected random rotation angles with nonzero means to evaluate the method in a more challenging

case. The comparison results for the case of dominant forward translation can be seen in Fig. 3.6a and Fig. 3.6b. In Fig. 3.7a and Fig. 3.7b, the translation and rotation errors for the case of dominant side translation are visualized. We can see that the regularization constraint improved the translation estimation slightly; however, it resulted in noticeable improvements for the rotation estimation in case of forward translation and good improvements for side translation.

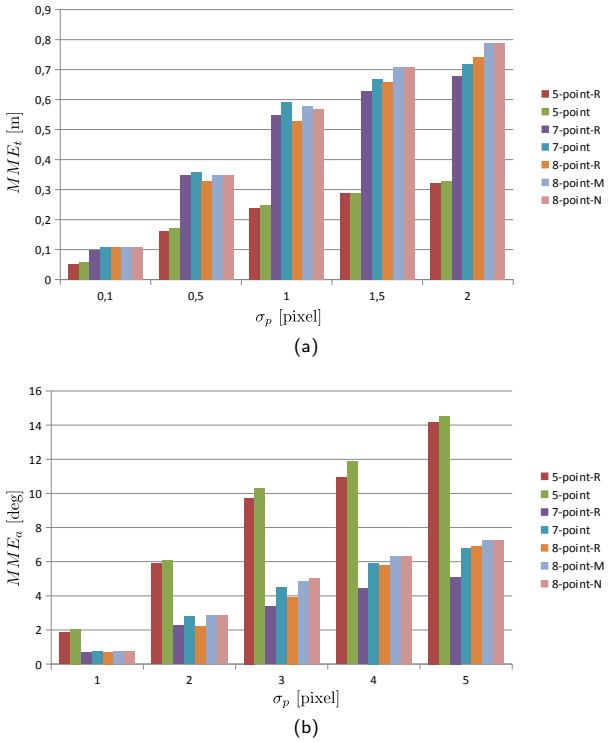


Figure 3.7: Mean of magnitudes of errors for translation in case of dominant side translations.

Surprisingly, we see that the 5-point method had a poor performance in rotation estimation (possibly due to numerical errors) in case of side motion. As a result, we can conclude that the 7-point-R method provides more reliable estimations if the camera experiences different types of motions.

3.5 Experimental Results

In this section, we first evaluate the performance of different methods for the estimation of the essential matrix (ego-motion up to scale) in subsection 3.5.1 based on the training sequences of the KITTI dataset for visual odometry [KIT15]. Then the estimation of scale of translations is taken into account in subsection 3.5.2 for the test sequences of the KITTI dataset.

3.5.1 Motion Estimation Up to Scale

To evaluate our proposed methods, we used the KITTI dataset for visual odometry [KIT15]. The dataset includes 22 long challenging sequences. The first eleven sequences are for training and their ground truths are provided. In table 3.3, the number of frames and the lengths of the training sequences are listed. As monocular camera motion can generally be estimated up to scale and absolute scale of translations is unknown if no other source of information is used, in this experiment, we replaced the scale factor from the provided ground truths of the first eleven sequences and conducted the evaluation based on the new ground truths. We applied the proposed regularization term for the 8-point, 7-point and 5-point methods and compared them with the original methods. The methods were implemented using C++ and ran on a computer with an Intel(R) Core(TM)2 Duo 3.33 GHz CPU. All the methods were run in the context of a random sample consensus algorithm (RANSAC), in which several essential matrices based on randomly selected N-matched points were calculated and then the best essential matrix which defined the flow of all points with the minimum epipolar distance errors was selected.

For quantitative comparisons, we used two types of accumulating errors: first, the mean of magnitudes of errors between all estimated and ground truth camera poses denoted as MME_p. If the camera position at time k is $\mathbf{c}_k$ and the estimates position is $\hat{\mathbf{c}}_k$, $MME_{\mathbf{c}}$ will be:

$$MME_{\mathbf{c}} = \frac{1}{K} \sum_{k=1}^{K} \sqrt{\epsilon_{\mathbf{c},k}^T \epsilon_{\mathbf{c},k}} \tag{3.39}$$

Table 3.3: The number of frames and the lengths of training sequences of the KITTI dataset for visual odometry.

Sequence	00	01	02	03	04	05
#Frames	4541	1101	4661	801	271	2761
Length [m]	3723.6	2453.1	5067.0	560.9	393.7	2205.5
Sequence	06	07	08	09	10	Avg.
#Frames	1101	1101	4701	1591	1201	2109.2
Length [m]	1231.5	694.7	3222.5	1705.0	919.4	2016.1

where $\epsilon_{\mathbf{c},k} = \hat{\mathbf{c}}_k - \mathbf{c}_k$.

The second measure is the mean of magnitudes of errors between the estimated and ground truth angles at all poses: MME_a. If the orientation of camera at time k is shown by the rotation matrix R_k and the estimated camera orientation is $\hat{R}_k$, MME_a will be:

$$MME_a = \frac{1}{K}\sum_{k=1}^{K}\epsilon_{a,k} \tag{3.40}$$

where $\epsilon_{a,k} = \frac{180}{\pi}\sqrt{3 - trace(R_{e,k})}$ and $R_{e,k} = R_k^T\hat{R}_k$.

For feature tracking, we used Harris corner detectors and forward backward Lucas-Kanade optical flow proposed in section 3.1.2 by using OpenCV library [Ope15]. The comparison results can be seen in table 3.4 and table 3.5.

Clearly, applying the regularization constraint improves the original 8- and 7-point methods significantly for all sequences and improves the 5-point method for some sequences. To have a better perception of the enhancements achieved through applying regularization terms for different methods, we visualized the estimated paths using the 8-point-N and the 8-point-R for the sequences 0, 1 and 9 along with the errors at each frame in Fig. 3.12 to Fig. 3.14. We can observe that the regularization term prohibited large rotation errors and consequently, the estimated paths are very close to the ground truth paths.

The camera poses are with respect to a coordinate system attached to the first camera pose in each sequence. In Fig. 3.8, the alignment of the axis of the mentioned coordinate system is depicted.

Table 3.4: Mean of magnitudes errors between the estimated and ground truth paths. Suffix (-R) is for regularized methods. 8-point-M is the modified 8-point method based on the weighted coplanarity constraints. 8-point-N is the normalized method proposed in [Har97].

Sequence	00	01	02	03	04	05
	$MME_{\mathbf{c}}$[m/frame]					
7-point-R	23.4	25.0	12.8	2.4	0.9	22.9
7-point	32.3	139.3	30.0	4.7	1.5	27.4
5-point-R	9.2	53.5	24.0	3.0	1.1	30.1
5-point	9.4	78.7	36.3	3.6	1.3	30.1
8-point-R	25.3	40.7	47.3	6.2	1.6	28.6
8-point-M	70.3	126.2	93.3	11.2	1.6	67.3
8-point-N	70.0	443.5	97.4	23.6	5.3	66.5

Sequence	06	07	08	09	10	Avg.
	$MME_{\mathbf{c}}$[m/frame]					
7-point-R	5.5	12.2	58.7	11.2	8.3	11.6
7-point	10.5	14.9	61.5	25.6	12.0	30.1
5-point-R	8.3	10.5	58.0	12.0	9.2	19.9
5-point	8.8	10.8	67.3	16.6	9.4	24.75
8-point-R	13.0	13.8	67.6	16.2	14.8	25.0
8-point-M	10.2	22.8	82.9	44.0	24.3	50.4
8-point	27.3	22.7	124.9	42.0	38.6	87.4

To have an impression of the contents of different sequences, in Fig. 3.9, Fig. 3.10 and Fig. 3.11, some of sample frames from the sequences 0, 1 and 9 are presented. In Fig. 3.15, Fig. 3.16, Fig. 3.17, the estimated paths and error charts for the sequences 0, 1 and 9 are depicted. We can see that the regularization term also resulted in better estimations. Finally, the comparison between the estimated paths for the sequences 0, 1 and 9 is presented in Fig. 3.18, Fig. 3.19 and Fig. 3.20.

In average, the 7-point-R method yielded the best results. To understand the results better, it should be mentioned that in the estimation of essential matrices, the most challenging case is when the base line motion is small and the rotations are high. It can be verified that in such cases the coplanarity equations tend to zero regardless of the motion parameters. Hence, in these cases, measurement noise can give rise to large errors in the estimation of essential matrices and motion parameters (especially the rotation parameters). These situations occur very often in the KITTI dataset when the car drives trough sharp bends. Consequently, the 8-point methods had a very poor performance almost in all

Table 3.5: Mean of magnitudes of errors between the estimated and ground truth angles. Suffix (-R) is for regularized methods. 8-point-M is the modified 8-point method based on the weighted coplanarity constraints. 8-point-N is the normalized method proposed in [Har97].

sequence	00	01	02	03	04	05
	MME_a[deg/frame]					
7-point-R	4.58	3.40	3.56	1.16	0.53	2.71
7-point	8.07	22.95	7.37	1.12	0.75	4.63
5-point-R	1.66	7.43	5.49	1.28	0.54	5.03
5-point	1.66	9.41	7.38	1.24	0.42	5.53
8-point-R	2.59	8.72	9.07	2.02	0.39	3.47
8-point-M	9.03	56.57	9.24	2.13	1.30	17.31
8-point-N	15.80	82.65	7.38	4.49	9.29	10.46

sequence	06	07	08	09	10	Avg.
	MME_a[deg/frame]					
7-point-R	4.33	4.23	1.83	2.54	1.32	2.74
7-point	5.82	5.63	3.45	4.98	6.74	6.50
5-point-R	2.97	3.58	1.51	1.04	2.27	2.98
5-point	2.92	3.90	1.61	2.53	1.64	3.48
8-point-R	5.89	6.26	3.91	3.85	4.60	4.62
8-point-M	16.93	9.53	12.05	11.45	11.00	14.23
8-point-N	8.65	5.53	16.11	11.37	12.16	16.72

of the sequences. On the contrary, applying the regularization constraint made the 8-point-R method have almost the same performance as the 7-point method at a much lower elapsed time which shows that it could be a proper option for real-time applications.

The 5-point-R method, as expected, had a better performance than the 7-point method and even for the sequence 0, it outperformed 7-point-R. But for the other sequences, it performed either similar to or worse than the 7-point-R (e.g. sequence 1). We analyzed the sequences more precisely and noticed that in the sequences where the outlier ratio is high, the 5-point method works more robust than other methods. The reason lies in the RANSAC part, in which N-matched points are selected randomly. Clearly, the less N matched points are selected, the more it is possible that the matched points do not contain any outliers. On the other hand, based on the simulation results, the 5-point-R method does not perform well to estimate rotation matrices in case of dominant side motions (occurred in sharp bends) which explains why it could not outperform the 7-point-R method.

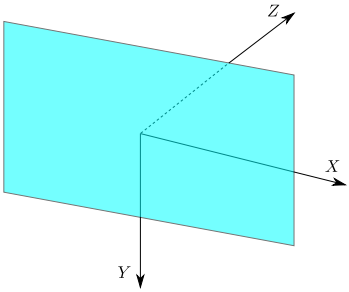


Figure 3.8: Coordinate system aligned to the image plane of a camera.

The elapsed time for different methods can be seen in table 3.6. We see that the inclusion of the regularization term increased the elapsed time which was the cost to increase the quality of the estimation of motion parameters. According to this table, for real-time purposes, the 8-point-R method is an appropriate option since it is not only fast but also estimate camera motions much better than normalized 8-point method (8-point-N).

Table 3.6: Average elapsed time (E. T.) per frame for different methods. 5PR= 5-point-R, 5p= 5-point, 7PR= 7-point-R, 8PR= 8-point-R, 8PM= 8point-M and 8PN= 8-point-N.

Method	5PR	5P	7PR	7P	8PR	8PM	8PN
E. T. [s/frame]	4.82	4.12	0.39	0.32	0.21	0.12	0.13

(a) Frame 0

(b) Frame 20

(c) Frame 40

(d) Frame 60

Figure 3.9: Sample frames from sequence 0.

(a) Frame 0

(b) Frame 20

(c) Frame 40

(d) Frame 60

Figure 3.10: Sample frames from sequence 1.

(a) Frame 0

(b) Frame 20

(c) Frame 40

(d) Frame 60

Figure 3.11: Sample frames from sequence 9.

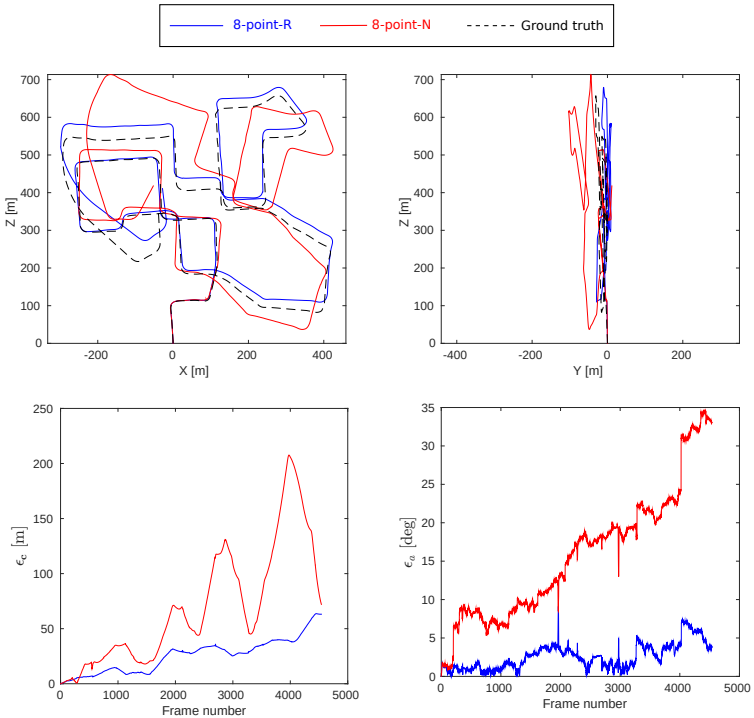


Figure 3.12: Estimation of camera poses for the sequence 0 of the KITTI dataset using the 8-point and 8-point-R methods. Top: estimated paths and ground truth. Bottom: errors of camera position and orientation at each frame. Clearly, the 8-point-R method managed to estimate camera poses near to the ground truth, while the 8-point method yielded a poor estimation. The errors are mainly originated from the poor estimations of rotation matrices when the car drives through sharp bends. In this cases, many of the points will have large coordinates resulting in the amplification of measurement noise significantly.

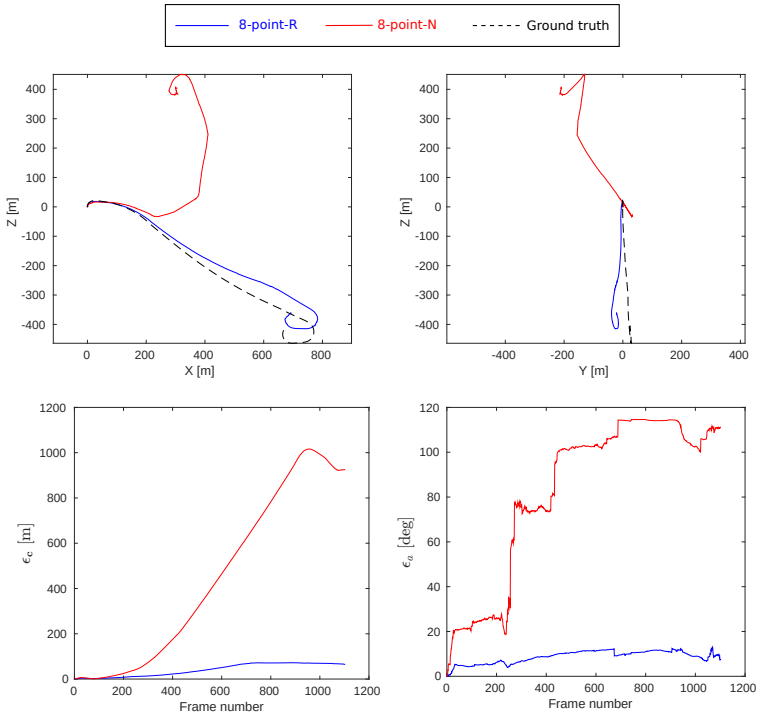


Figure 3.13: Estimation of camera poses for the sequence 01 of the KITTI dataset using the 8-point and 8-point-R methods. Top: estimated paths and ground truth. Bottom: errors of camera position and orientation at each frame. In this sequence, the car drives in an Autobahn, where most features are far from the camera. It results in small feature displacements subjected to measurement noise more (small signal to noise ratio). Additionally, due to the repeated patterns of guardrails, the number of wrong matches (outliers) is relatively high. These two issues give rise to large errors in estimation of rotation matrices so that the estimated path using the 8-point method is totally wrong. Whereas, thanks to the regularization term, the 8-point-R method was able to deal with the disturbances effectively.

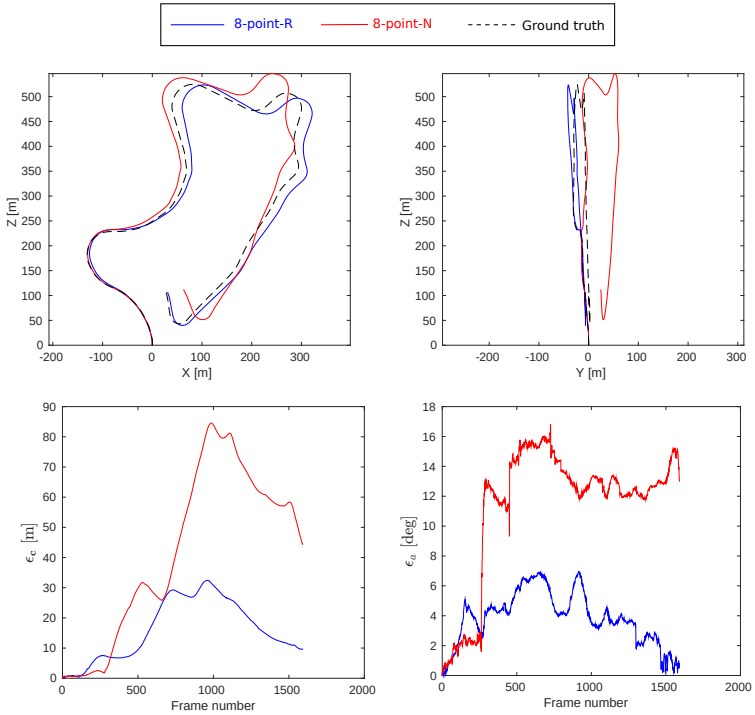


Figure 3.14: Estimation of camera poses for the sequence 09 of the KITTI dataset using the 8-point and 8-point-R methods. Top: estimated paths and ground truth. Bottom: errors of camera position and orientation at each frame. The results for this sequence depict the good performance of the 8-point-R method in the estimation of pitch and roll angles (ϕ and ψ). Since the 8-point method does not take into account the dependencies of the essential matrix elements, it generally has a poor performance in the estimation of pitch and roll angles.

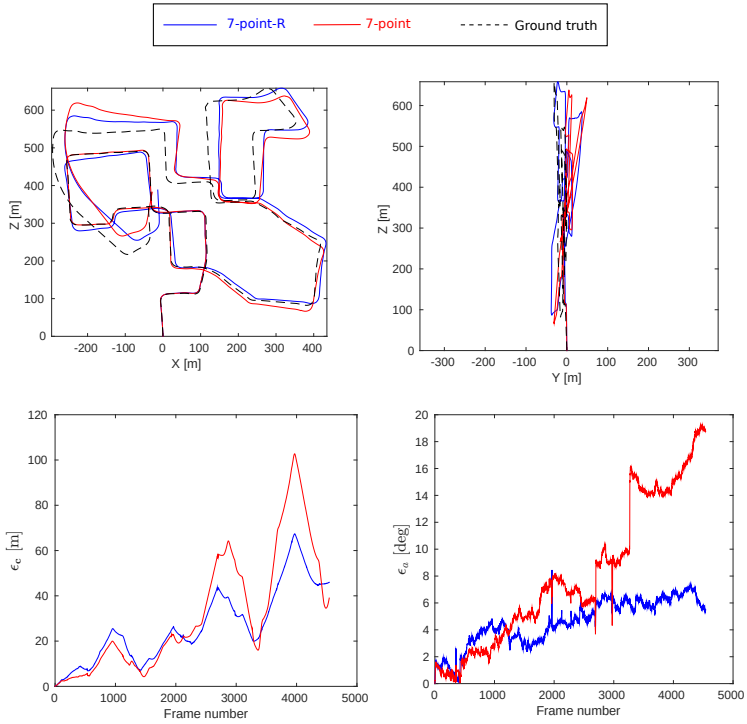


Figure 3.15: Estimation of camera poses for the sequence 0 of the KITTI dataset using the 7-point and 7-point-R methods. Top: estimated paths and ground truth. Bottom: errors of camera position and orientation at each frame. In this sequence, the 7-point-R method was able to estimate rotation matrices in the sharp bends better than the 7-point method.

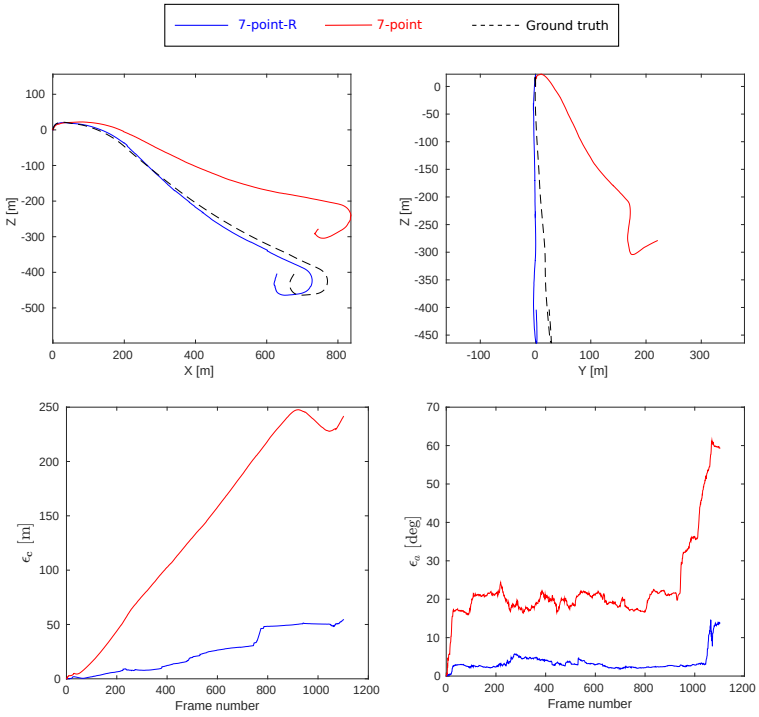


Figure 3.16: Estimation of camera poses for the sequence 01 of the KITTI dataset using th 7-point and 7-point-R methods. Top: estimated paths and ground truth. Bottom: errors of camera position and orientation at each frame. The 7-point method could not deal with the high ratio of outliers effectively, resulting in large errors in estimation of the essential matrices. Whereas, the 7-point-R method could estimate the path very close to the ground truth.

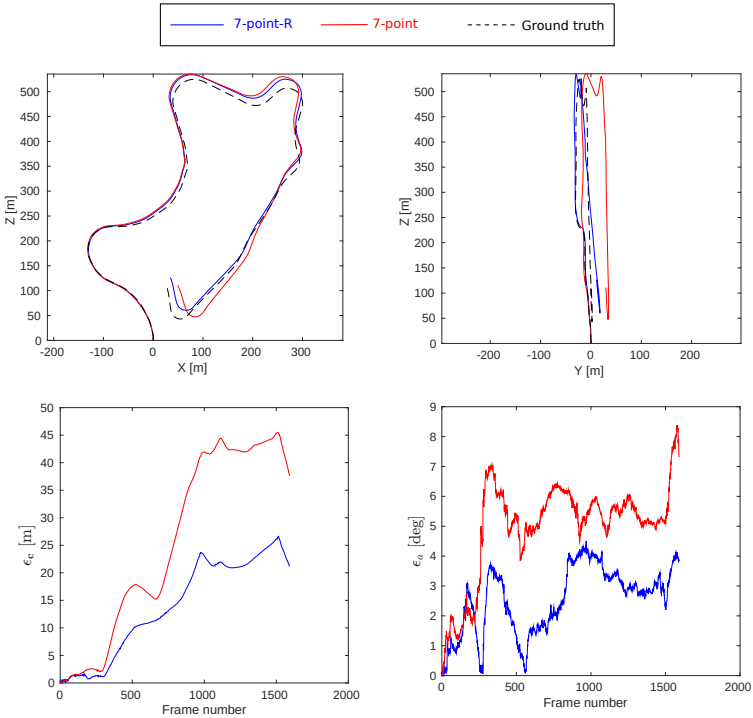


Figure 3.17: Estimation of camera poses for the sequence 09 of the KITTI dataset using th 7-point and 7-point-R methods. Top: estimated paths and ground truth. Bottom: errors of camera position and orientation at each frame. It can be seen that the 7-point-R method has a better performance in estimation of roll and pitch angles, resulting in better estimation of the elevation of the car.

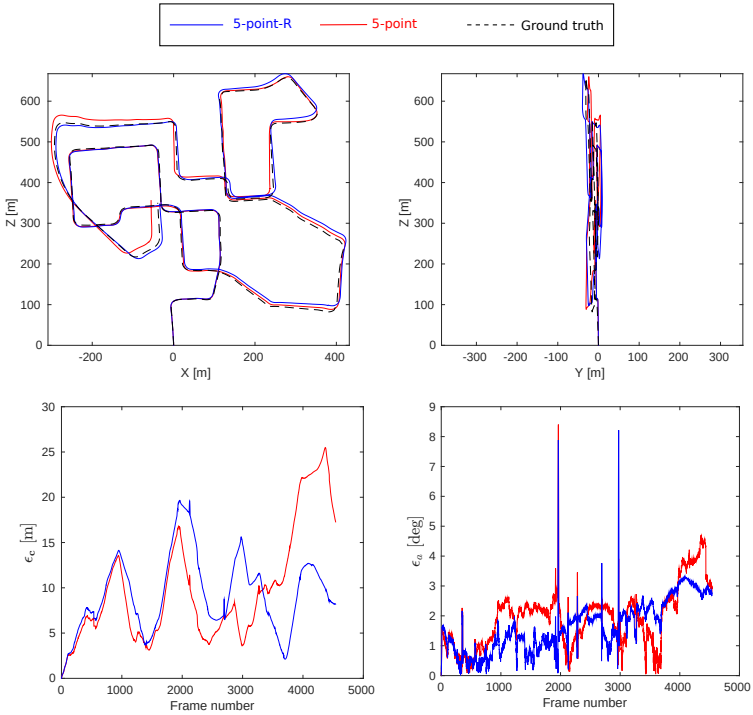


Figure 3.18: Estimation of camera poses for the sequence 0 of the KITTI dataset using the 5-point and 5-point-R methods. Top: estimated paths and ground truth. Bottom: errors of camera position and orientation at each frame. Both methods yielded very good estimations of motion parameters. However, a slight better performance of the 5-point-R method in the estimation of camera poses is also visible.

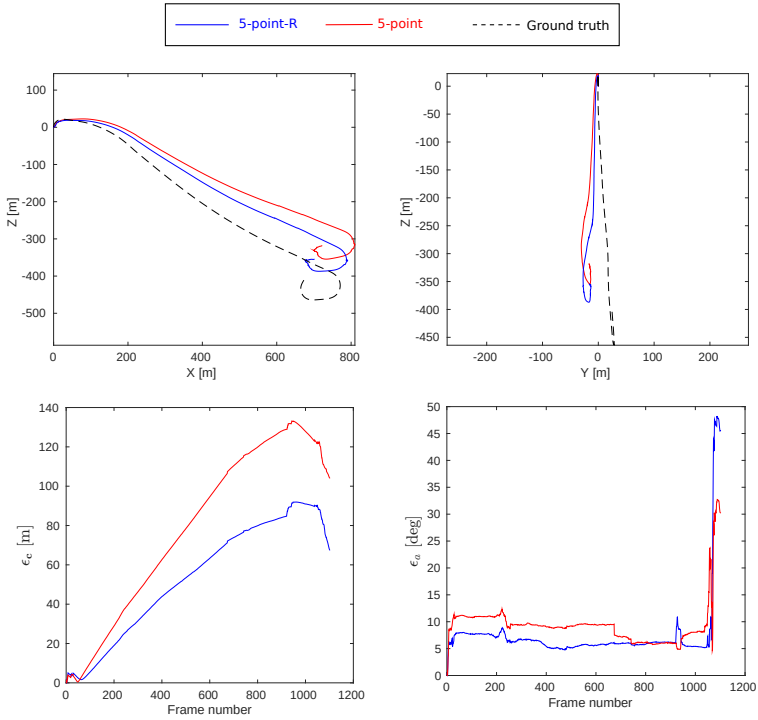


Figure 3.19: Estimation of camera poses for the sequence 01 of the KITTI dataset using th 5-point and 5-point-R methods. Top: estimated paths and ground truth. Bottom: errors of camera position and orientation at each frame. The 5-point(-R) methods deal with outliers effectively since they use less number of matched points. But in this sequence, they performed poorly in comparison to the 7-point-R method, showing the sensitivities of the 5-point methods to relatively high measurement noises.

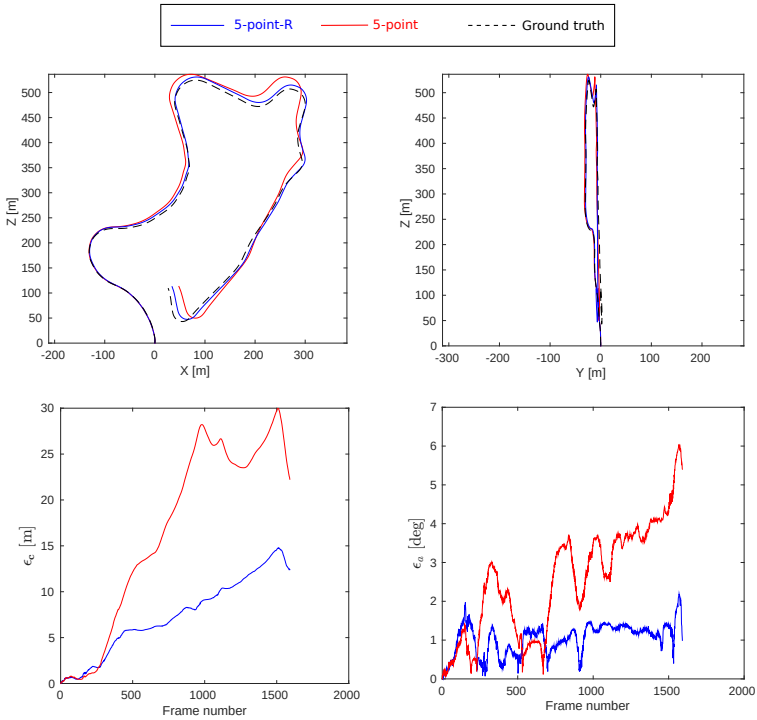


Figure 3.20: Estimation of camera poses for the sequence 09 of the KITTI dataset using th 5-point and 5-point-R methods. Top: estimated paths and ground truth. Bottom: errors of camera position and orientation at each frame. The 5-point method had a slight better estimation of yaw angles (θ).

3.5.2 Absolute Scale Detection

If a camera is installed on a wheeled vehicle, an intuitive method to detect scale of translations is to use the height of the camera over the ground plan as a known parameter. This method has been used in [GZS11], [SCG13] and [SC14]. To utilize this parameter, it is necessary to track features on the ground plane. Nevertheless, tracking features belonging to ground planes is challenging since typically ground planes are highly homogeneous and the feature are not distinctive enough to be tracked uniquely. Such features cannot be tracked using the Lucas-Kanade optical flow method as it relies on the relatively high amount of brightness gradients about the feature points. Hence, feature matching or dense matching techniques should be utilized to track points on the ground planes. Among the feature matching methods, those with binary descriptors such as ORB or BRISK also fail in most cases as they cannot capture reliable binary patterns if a neighborhood is homogeneous. In such cases, a small amount of noise changes the binary patterns significantly. On the other hand, SIFT or SURF descriptors which are based on gradient patterns could be good variants; however, unfortunately, they are computationally expensive. In [GZS11], a simple descriptor with the length of eighteen based on the gradients about each feature in a window with the size 3×3 is utilized. This method works well if the homogeneity is not high. Hence, in [SC14], a dense matching over a region of interest between two frames was conducted, which yielded relatively good estimations of scale factors. However, this method is computationally expensive.

To deal with this problem, we push forward a new descriptor and a new matching technique which are concurrently fast and accurate. In our method, we extract corner features from two consecutive frames at different pyramid levels and assign a compact version of the SURF descriptor to them. The mentioned descriptor contains the average of image gradients in the four windows in the neighborhood of each feature. The length of the descriptor is eight, which results in a very fast matching process. We call this descriptor distributed averages of gradients (DAG). Given two frames, features in the i^{th} frame ($i = 1, 2$) is denoted as $\mathbf{f}_{j,i} = (x_{j,i}, y_{j,i})$ with the descriptors $\mathbf{d}_{j,i}$, where $j = 1...M_i$ and M_i is the number

of features in the i^{th} frame. The Euclidean distance between two descriptor vectors is denoted as:

$$e_{r,s} = ||\mathbf{d}_{r,2} - \mathbf{d}_{s,1}|| \tag{3.41}$$

For each feature in the second frame e.g. $\mathbf{f}_{r,2}$, two matches in the first frame with the minimum distances are found:

$$\begin{aligned} &(\mathbf{f}_{r,2}, \mathbf{f}_{s,1}) \\ &(\mathbf{f}_{r,2}, \mathbf{f}_{t,1}) \end{aligned} \tag{3.42}$$

where $e_{r,s} < e_{s,t}$. Practically, we found that if $e_{r,s}/e(r,t) < 0.4$, it is highly probable that the first match is correct. Otherwise, both of the matched features can be potential candidates for the correct match. One important criterion to reject many of the outliers are the distances of the matched points to their correspondent epipolar lines. In the matching process if the candidate matched points have distances more than five pixels, we do not consider them as possible matches. In Fig. 3.21, tracking of low quality features on the ground plane is visualized. It can be seen hat the ground plane is highly homogeneous, but the method managed to track many features.

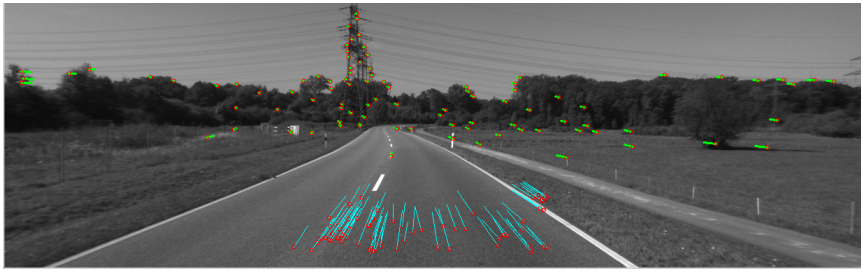


Figure 3.21: Tracking low quality features on the ground plane based on the DAG descriptors and epipolar constraint. The green lines represent feature tracking using FBLK and the blue lines represent feature tracking using DAG and the epipolar constraint.

After finding two possible matches for each feature in the second frame, we consider both possibilities and calculate the scale factor for both cases. Given N features in the second frame, more than N scale factors are calculated. Obviously, the best scale factor is the one which is more repeated. To find the most repeated scale factor, a median filter can be used. To this end, all the obtained scale factors are sorted and then the most probable scale factor will be the middle value.

The above procedure should be run if the number of tracked features on the ground plane is less than a threshold. We used the proposed method for scale determination and submitted the results to the KITTI portal under the name RCMPE+GP (relative camera pose estimation + ground plane). In table 3.7, the ranking of all submitted methods to the KITTI portal on the date 15.07.2015 is presented. Not surprisingly, the best estimations are delivered using the point cloud data (V-LOAM) [ZS15]. These methods require high computation load. Additionally, point cloud data is produced by a 3D laser scanner (LIDAR) which is very expensive and heavy. Following, the methods based on LIDAR, stereo vision based methods are ranked. We notify that both of the LIDAR and stereo vision based methods enjoy range and bearing measurements, while in monocular methods, only bearing measurements are utilized. Nevertheless, it can be seen that our method outperforms many of the stereo vision based methods.

Among the published monocular methods, our method has the rank second after the MLM-SFM method [SC14]. In Fig. 3.22, the average of translation errors and the average of rotation errors at different distances of the paths are depicted. We can see that libviso method [GZS11] has large drifts both in the estimations of rotation matrices and translation vectors. In average, our method has 2.55% error in the estimation of translation and the MLM-SFM method has 2.54% error. It should be considered that MLM-SFM is based on the multiple observations of features and iterative optimization of a set of consecutive camera poses. For the scale detection, MLM-SFM uses dense plane matching and a Kalman filter to estimate scale more smoothly. Furthermore, this method uses extra processes to detect moving objects and remove features on the moving objects. Whereas, in our method, we achieved good estimations based on only two consecutive frames and sparse matching of features on the ground plane.

From the eleven submitted sequences, $X - Z$ plots of the first five sequences are visualized in the KITTI Portal. In Fig. 3.23 and Fig. 3.24, the estimated paths based on our method (RCMPE+GP), MLM-SFM and libviso are depicted. Visually, it can be seen that our method is near to the ground truths in sequences 11, 13 and 15. MLM-SFM has better estimation for the sequence 12. The better performance of the MLM-SFM method lies in the fact that the method uses a separate process to detect features on moving objects. As a result, it has better performance in the sequences in which many moving objects (cars) exist.

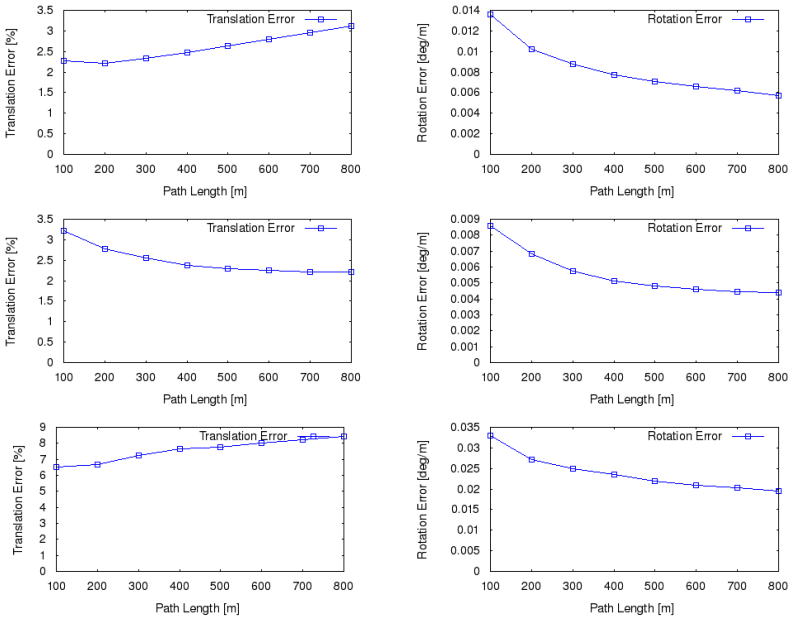


Figure 3.22: Averages of translation and average of rotation errors for the test sequences of KITTI dataset: RCMPE+GP (top), MLM-SFM (middle) and libviso (bottom). The images are adapted from [KIT15].

Table 3.7: Ranking of all methods in the KITTI website on 15.07.2015. pc: point cloud, st: stereo, m: monocular.

Rank	Method	data	T. error [%]	R. error [deg/m]	Published
1	VLOAM	pc	0.75	0.0018	yes
2	LOAM	pc	0.88	0.0022	yes
3	SOFT	st	1.03	0.0029	no
4	cv4xv1-sc	st	1.09	0.0029	yes
5	DEMO	pc	1.14	0.0049	yes
6	MFI	st	1.3	0.003	yes
7	TLBBA	st	1.36	0.0038	yes
8	2FO-CC	st	1.37	0.0035	yes
9	VoBa	st	1.46	0.003	no
10	MuProV	st	1.5	0.0041	no
11	StereoSFM	st	1.51	0.0042	yes
12	JDO	st	1.55	0.0039	no
13	SSLAM	st	1.57	0.0044	yes
14	Stereo VO	st	1.59	0.0033	no
15	BackwardVO	st	1.6	0.0036	no
16	MVO	m	1,6	0.0029	no
17	BA-MFT	st	1.62	0.003	no
18	eVO	st	1.76	0.0036	yes
19	SOVI	st	1.8	0.0079	no
20	D6DVO	st	2.04	0.0051	yes
21	MICP_VO	st	2.13	0.0065	no
22	SSLAM-HR	st	2.14	0.0059	yes
23	W-SFM	m	2.16	0.0033	no
24	FTMVO	m	2.24	0.0049	no
25	NOSM	st	2.28	0.004	no
26	LCMVO	m	2,33	0.005	no
27	VISO2-S	st	2.44	0.0114	yes
28	MLM-SFM	m	2.54	0.0057	yes
29	GT_VO3pt	st	2.54	0.0087	yes
30	BoofCV-SQ3	st	2.54	0.0073	no
31	RCMPE+GP	m	2.55	0.0086	yes
32	VO3pt	st	2.69	0.0068	yes
33	CDR	st	2.75	0..0045	no
34	TGVO	st	2.94	0.0077	yes
35	FA	m	2.98	0.0073	no
36	SVO	st	3.01	0.008	no
37	VO3ptLBA	st	3.13	0.0104	yes
38	MSD_VO	st	3.57	0.0109	no
39	CFORB	st	3.73	0.0107	no
40	VOFS	st	3.94	0.0099	yes
41	VOSLABA	st	4.17	0.0112	yes
42	CFROB	st	5.32	0.0128	no
43	VISO2-M+GP	m	7.46	0.0245	yes
44	RMVO	m	8.33	0.0233	no
45	LPF	m	8.98	0.0246	no
46	VISO2-M	m	11.94	0.0234	yes
47	OABA	m	20.95	0.0135	no

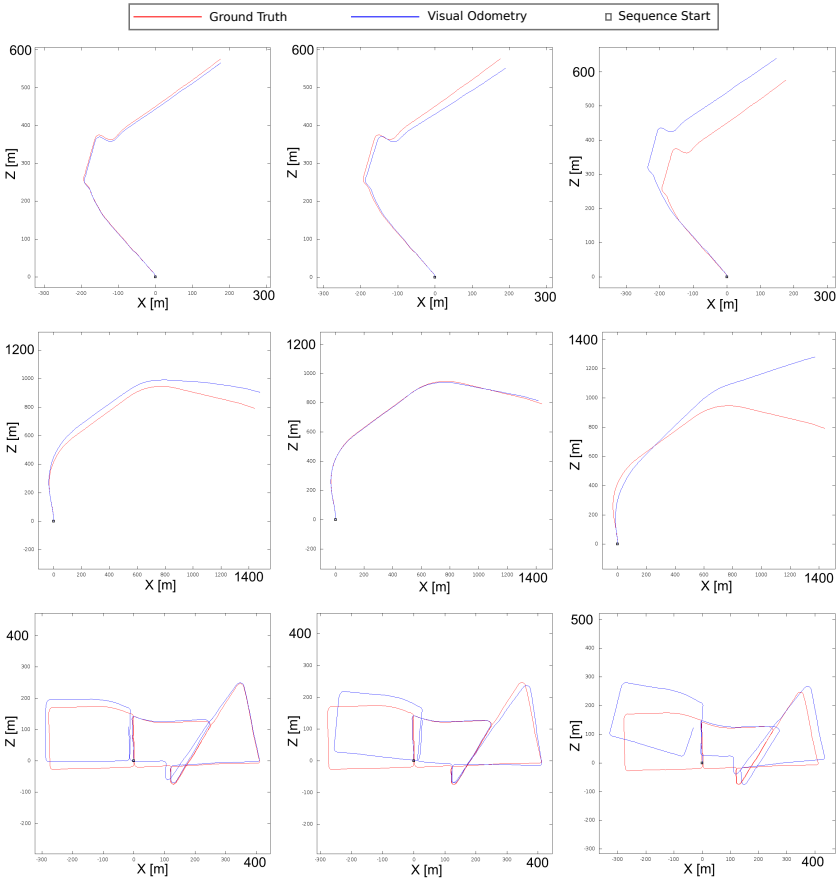


Figure 3.23: Estimated $X - Z$ paths using different methods: RCMPE+GP (left column), MLM-SFM (middle column) and libviso (right column).The images are adapted from [KIT15].

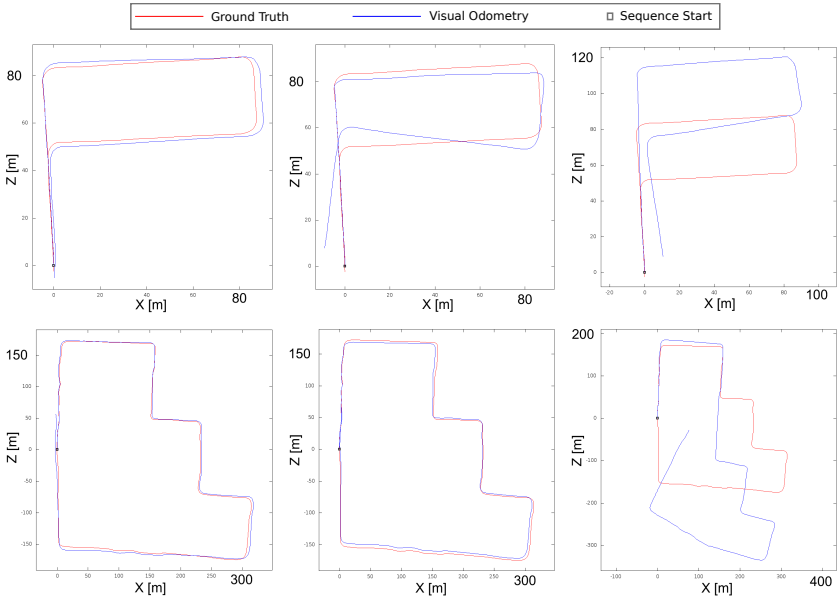


Figure 3.24: Estimated $X - Z$ paths using different methods: RCMPE+GP (left column), MLM-SFM (middle column) and libviso (right column). The images are from [KIT15].

4 2D Monocular SLAM

In this chapter, new techniques are proposed to localize mobile robots and create maps of landmarks simultaneously using a monocular camera. To this end, unlike the visual odometry methods, we leverage multiple observations of landmarks. Multiple observations generally enhance the estimation of robot and landmark positions as the effect of measurement noise is suppressed more. In the case of monocular (bearing only) SLAM, a natural way to consider the uncertainty of a landmark at the initialization time is to assume a uniform distribution over a ray at which the landmark is observed for the first time. This distribution gradually tends towards a Gaussian distribution if the landmark is observed again at more parallax angles. Nevertheless, we can consider a line segment as a support for these distributions of which ending points have direct relations with the 2σ confidential region of a Gaussian distribution or a uniform distribution. In this chapter, we discuss how to use line segments to model the uncertainties in the positions of landmarks in the 2D monocular SLAM problem. This chapter is based on our works in [MM12b], [MM12c] and [MM13a].

4.1 A Particle Filter Based Method

As illustrated in chapter 2.3, the main problem of bearing only SLAM is how to handle the large uncertainties of landmarks. Considering Fig. 4.1, in the presence of measurement noises $v_t \sim \mathcal{N}(0, \sigma_v)$, if a robot at the position $\mathbf{x}_t^R$ observes the landmark L_j at the bearing angle ϕ_j, the landmark position is distributed over a trapezoidal support. However, since measurement noise is a relative quantity, we can consider measurement noise at the initialization time zero, and consequently

the distribution will be simply on a line segment. Therefore, the distribution of a landmark at the initialization time is:

$$p(\mathbf{x}_j^L|\mathbf{x}_t^R, \phi_{j,t}) = \begin{cases} 1/(r_{max} - r_{min}), & \text{if} \quad c \\ 0, & \text{else} \end{cases} \tag{4.1}$$

where

$$c : y_j^L = m_{j,t}\, x_j^L + b_{j,t}\,\& \\ r_{min}\cos\psi_{j,t} + x_t^R < x_j^L < r_{max}\cos\psi_{j,t} + x_t^R$$

$\psi_{j,t} = \phi_{j,t} + \theta_t^R$, $m_{j,t} = \tan\psi_{j,t}$, $b_{j,t} = y_t^R - \tan(\psi_{j,t})x_t^R$. r_{min}, r_{max} are parameters which state the minimum and maximum ranges in which we expect the landmark exists. it means that we assume a uniform distribution for a landmark on a line segment in a predefined range given the position of the robot. A line segment can be represented as a function such as Γ as follows:

$$\Gamma(\mathbf{x}_{j,c,t}^L, \lambda_{j,t}, \alpha_{j,t}) \tag{4.2}$$

where $\mathbf{x}_{c,t}^L = (x_{c,t}^L, y_{c,t}^L)$ is the center of the line segment with the length λ_t and the angle $\alpha_{j,t}$ with respect to the X axis. Consequently, we can denote the line segment for the j^{th} landmark as $\Gamma_{j,t}$ which is $\Gamma_{j,t} = \Gamma(\mathbf{x}_{j,c,t}^L, \lambda_{j,t}, \alpha_{j,t})$.

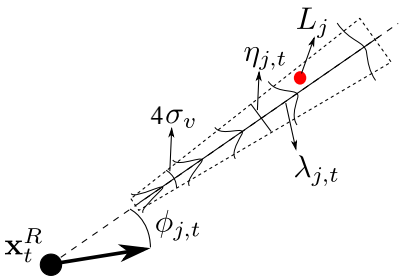


Figure 4.1: Landmark initialization using a trapezoid.

After the initialization of a landmark, the goal is to minimize the uncertainties of the robot and the landmark positions while the robot moves. The main idea is that the bounded uncertainty of the landmark results in the bounded uncertainty of the robot pose and vice-versa. Such a mutual relation in EKF based methods is implemented using a covariance matrix. In the following, considering the proposed model for a landmark uncertainty, such a relation in the context of a probability density function will be derived.

First, we start with the depth estimation of a feature if at least two different bearing observations of the feature are available. We know that if the landmark L_j is observed at two angles $\phi_{j,1}$ and $\phi_{j,2}$ at two different robot poses $\mathbf{x}_1^R$ and $\mathbf{x}_2^R$, the landmark is located on the intersection of the lines along which the landmark is observed, meaning:

$$\begin{aligned} y_j^L &= m_{j,1}x_j^L + b_{j,1} \\ y_j^L &= m_{j,2}x_j^L + b_{j,2} \end{aligned}$$

Solving the above equation system results in:

$$\begin{aligned} x_j^L &= (b_{j,2} - b_{j,1})/(m_{j,1} - m_{j,2}) \\ y_j^L &= m_{j,1}x_j^L + b_{j,1} \end{aligned}$$

If we consider measurement noise, the landmark should be localized on a line segment created by the intersection of a line segment and a trapezoid (Fig. 4.2). Clearly the distribution of the intersection points on the line segment depending on the parallax angle will not be uniform any more, but the ending points are always equivalent to the $2\sigma_v$ borders of a Gaussian distribution. As a result we can always keep the support of the distribution over a line segment.

The above calculation is valid if and only if the exact positions of the robot are available. Otherwise, minor errors in the robot pose can cause major errors in the calculation of the landmark positions, especially if the landmarks are observed at

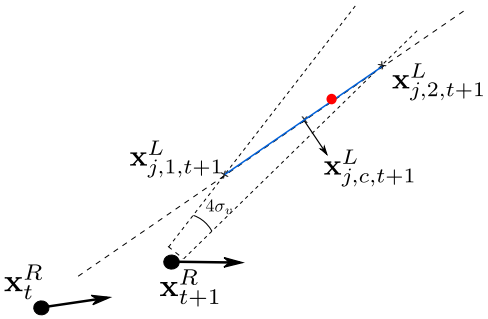


Figure 4.2: Trimming of a line segment in the presence of measurement noises.

low parallax angles. However, the intersecting method can be used in the context of a recursive Bayesian filter which includes prediction and update phases. Now, the problem is the estimation of the robot and the landmark positions, given the following measurement sequence:

$$\{\Phi_t\} = \{\Phi_{1,t}, ..., \Phi_{M,t}\}$$

where $\Phi_{j,t} = \{\phi_{j,t}, \phi_{j,t-1}, ..., \phi_{j,0}\}$, $j = 1...M$.

Speaking in probabilistic terms, if the vectors $\mathbf{x}^R_t = [x^R_t \ y^R_t \ \theta^R_t]^T$ and $\mathbf{x}^L_j = [x^L_j \ y^L_j]^T$ are defined, the following PDF (probability density function) should be estimated:

$$p(\mathbf{x}^R_t, \{\mathbf{x}^L_{j,t}\}|\{\Phi_t\}) \tag{4.3}$$

We first decompose Eq. (4.3) using the conditional probability rules as follows:

$$p(\mathbf{x}^R_t, \{\mathbf{x}^L_{j,t}\}|\{\Phi_t\}) \quad = \quad p(\{\mathbf{x}^L_{j,t}\}|\,\mathbf{x}^R_t, \{\Phi_t\})p(\mathbf{x}^R_t|\{\Phi_t\}) \tag{4.4}$$

Given Eq. (4.4), the recursive algorithm can be formed as follows:
Prediction:

$$p(\mathbf{x}^R_{t+1}|\{\Phi_t\}) = \int p(\mathbf{x}^R_{t+1}|\mathbf{x}^R_t, \mathbf{u}_t)p(\mathbf{x}^R_t|\{\Phi_t\})p(\mathbf{u}_t)d\mathbf{x}^R_t \, d\mathbf{u}_t \tag{4.5}$$

where $\mathbf{u}_t = [u_{f,t}, u_{r,t}]$.

Update:

$$p(\mathbf{x}_{t+1}^R|\{\Phi_{t+1}\}) = \frac{p(\{\phi_{t+1}\}|\mathbf{x}_{t+1}^R, \{\Phi_t\})}{p(\{\phi_{t+1}\}|\{\Phi_t\})} p(\mathbf{x}_{t+1}^R|\{\Phi_t\}) \tag{4.6}$$

Since the positions of the landmarks are independent of each other, Eq. (4.6) becomes:

$$p(\mathbf{x}_{t+1}^R|\{\Phi_{t+1}\}) = \frac{\prod_{j=1}^{M} p(\phi_{j,t+1}|\mathbf{x}_{t+1}^R, \Phi_{j,t})}{\prod_{j=1}^{M} p(\phi_{j,t+1}|\Phi_{j,t})} \; p(\mathbf{x}_{t+1}^R|\{\Phi_t\}) \tag{4.7}$$

where

$$p(\phi_{j,t+1}|\Phi_{j,t}) = \int p(\phi_{j,t+1}|\, \mathbf{x}_{t+1}^R, \Phi_{j,t}) p(\mathbf{x}_{t+1}^R|\Phi_{j,t}) d\mathbf{x}_{t+1}^R \tag{4.8}$$

and

$$p(\phi_{j,t+1}|\mathbf{x}_{t+1}^R, \Phi_{j,t}) = \int p(\phi_{j,t+1}|\, \mathbf{x}_{t+1}^R, \mathbf{x}_{j,t}^L) p(\mathbf{x}_{j,t}^L|\, \mathbf{x}_t^R, \Phi_{j,t}) p(\mathbf{x}_t^R|\Phi_{j,t})\, d\mathbf{x}_{j,t}^L\, d\mathbf{x}_t^R \tag{4.9}$$

The term $p(\mathbf{x}_{j,t}^L|\mathbf{x}_t^R, \Phi_{j,t})$ in Eq. (4.9) can be presented by a distribution located over a line segment which starts at the robot position and has a direction with the angle $\theta_t^R + \phi_{j,t}$ and is bounded by the border of the last estimation of $\mathbf{x}_{j,t}^L$ (Fig. 4.3).

The next step is to update the PDF of $\mathbf{x}_j^L$, meaning:

$$p(\mathbf{x}_{j,t+1}^L|\Phi_{j,t+1}) = \int p(\mathbf{x}_{j,t+1}^L|\mathbf{x}_{t+1}^R, \Phi_{j,t+1}) p(\mathbf{x}_{t+1}^R|\Phi_{j,t+1}) d\mathbf{x}_{t+1}^R \tag{4.10}$$

In Eq. (4.10), $p(\mathbf{x}_{j,t+1}^L|\mathbf{x}_{t+1}^R, \Phi_{j,t+1})$ again represents a distribution over a line segment which is limited by the border of $p(\mathbf{x}_{j,t}^L|\Phi_{j,t+1})$. This results in the required rule to trim the length of the line segments. However, the implementation

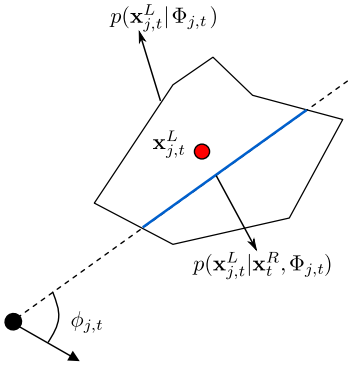


Figure 4.3: The possible landmark position given the robot pose and all the previous measurements.

of such an algorithm requires again to store the landmark regions as polygons and then find the intersection points, which gives rise to a very time consuming algorithm as discussed before. To ease this problem, we attempt to generalize the idea of using line segments attached to each robot pose which belongs to $p(\mathbf{x}^L_{j,t}|\Phi_{j,t+1})$ in order to represent part of the landmark uncertainty, but not necessarily along the line between the robot and the landmark. For the sake of simplicity, the initial angles of line segments, $\{\alpha_{j,t}\}$, are kept constant, and then the centers and lengths will be modified. Therefore, Eq. (4.10) is decomposed again as follows:

$$\begin{aligned} p(\mathbf{x}^L_{j,t+1}|\Phi_{j,t+1}) = \int & p(\mathbf{x}^L_{j,t+1}|\mathbf{x}^R_{t+1}, \Gamma_{j,t}, \mathbf{x}^R_t, \Phi_{j,t+1}) \times \\ & p(\mathbf{x}^R_{t+1}|\mathbf{x}^R_t, \Phi_{j,t+1}) p(\Gamma_{j,t}|\mathbf{x}^R_t, \Phi_{j,t+1}) \times \\ & p(\mathbf{x}^R_t|\Phi_{j,t+1})\, d\mathbf{x}^R_{t+1} d\mathbf{x}^R_t d\Gamma_{j,t} \end{aligned} \tag{4.11}$$

where $\Gamma_{j,t}$ is the line segment for the j^{th} landmark L_j. Considering the fact that $\Gamma_{j,t}$ can be uniquely determined when $\mathbf{x}^R_t$ and $\Phi_{j,t+1}$ are given, then $p(\Gamma_{j,t}|\mathbf{x}^R_t, \Phi_{j,t+1})$ will be a dirac-delta function of which integration will be one. Furthermore, in the first probability term, the conditions $(\mathbf{x}^R_t, \Phi_{j,t+1})$ do not give any more information if $\Gamma_{j,t}$ is available. Therefore, Eq. (4.11) becomes:

$$p(\mathbf{x}^L_{j,t+1}|\Phi_{j,t+1}) = \int p(\mathbf{x}^L_{j,t+1}|\mathbf{x}^R_{t+1}, \Gamma_{j,t}, \phi_{j,t+1}) \times p(\mathbf{x}^R_{t+1}|\mathbf{x}^R_t, \Phi_{j,t+1}) p(\mathbf{x}^R_t|\Phi_{j,t+1})\, d\mathbf{x}^R_{t+1} d\mathbf{x}^R_t \tag{4.12}$$

This decomposition gives us the possible locations of $\mathbf{x}^L_{j,t+1}$ over $\Gamma_{j,t}$. Obviously, based on distribution of the locations over the line segment, the centers and lengths of the updated line segments can be achieved. In Eq.(4.12), the term $p(\mathbf{x}^L_{j,t+1}|\mathbf{x}^R_{t+1}, \Gamma_{j,t}, \phi_{j,t+1}) p(\mathbf{x}^R_{t+1}|\mathbf{x}^R_t, \Phi_{j,t+1})$ has the interpretation that assuming a robot pose $\mathbf{x}^R_t$, a set of $\mathbf{x}^R_{t+1}$s denoted as $X^R_{t+1}|\mathbf{x}^R_t$ can be achieved. Then, for each $\mathbf{x}^R_{t+1} \in X^R_{t+1}$ and given $\phi_{j,t+1}$ a line segment can be assumed which intersects $\Gamma_{j,t}$. These intersected regions can be included again in new line segments in a way shown in Fig. 4.2 with the centers $\mathbf{x}^L_{j,c,t+1}|\mathbf{x}^R_{t+1}$. Based on the numerical simulation and also as we will show in section 4.2, we observe that the distribution of these centers would have a non-symmetric shape as shown in Fig. 4.4. As can be seen, the density of the centers decreases for the further locations on line segments. This issue should be considered when we want to sample the PDF of the robot and the landmark positions to work with particle filters. In this case, the lengths of the updated Γ should be determined in such a way that we do not lose any piece of data by which the algorithm may diverge.

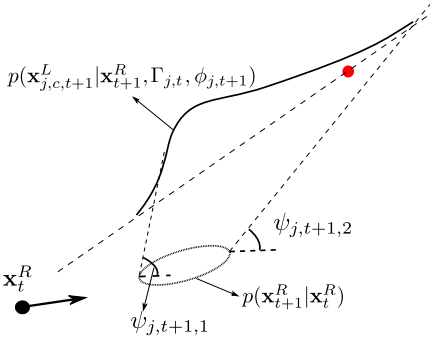


Figure 4.4: The PDF of new centers over the $\Gamma_{j,t}$. $\psi_{j,t+1,1} = \theta^R_{t+1} + \phi_{j,t+1} + 2\sigma_v + 2\sigma_{w2}$ and $\psi_{j,t+1,2} = \theta^R_{t+1} + \phi_{j,t+1} - 2\sigma_v - 2\sigma_{w2}$

4.1.1 Particle Filter Implementation

The recursive algorithm mentioned in the previous section can be implemented using a modified particle filter approach. In such an approach, the possible positions of the robot are assumed as a finite set of points in R^3 space, while the uncertainty of landmarks are modeled by a set of line segments Γ each of which is bound to a robot pose. Thus, the estimated vector will be $[\mathbf{x}_t^R, \{\Gamma_{j,t}\}]^T$.

Based on the derived formulation in the previous section, the following algorithm can be generated:

Prediction:

Generate N_p samples (particles) from the PDF $p(\mathbf{x}_t^R|\Phi_{j,t+1})$ such as $X_t^R = \{\mathbf{x}_{t,1}^R, ..., \mathbf{x}_{t,N_p}^R\}$, and the N_p attached line segments $(\Gamma_{j,t,i})$ for each landmark such as $X_{j,t}^L = \{\Gamma_{j,t,1}^L, ..., \Gamma_{j,t,N_p}^L\}$, where $j = 1, .., M$. Generate also N_p samples from $p(\tilde{\mathbf{u}}_t)$ such as $\tilde{U}_t = \{\tilde{\mathbf{u}}_{t,1}, ..., \tilde{\mathbf{u}}_{t,N_p}\}$. By plugging X_t^R and $\tilde{U}_t$ into the motion model (Eq. 2.79), the next possible robot poses $\hat{X}_{t+1}^R = \{\hat{\mathbf{x}}_{t+1,1}^R, ..., \hat{\mathbf{x}}_{t+1,N_p}^R\}$ are obtained.

Considering $X_{j,t}^L$ and $\hat{X}_{t+1}^R$, a predicted range of the measurement for each pair of a robot pose and a landmark position $(\hat{\mathbf{x}}_{t+1,r}^R, \Gamma_{j,t,r})$, $(r = 1...N_p)$, such as $\bar{\phi}_{j,t+1,r}$ is obtained (Fig. 4.5):

$$\bar{\phi}_{j,t+1,r} = [\hat{\phi}_{j,1,t+1,r}, \hat{\phi}_{j,2,t+1,r}] \tag{4.13}$$

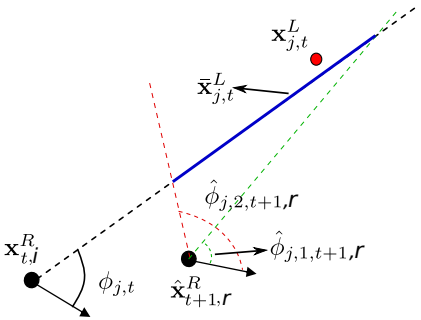


Figure 4.5: The expected range of the bearing measurement in the next step.

Update:
As soon as the new measurements $\{\phi_{j,t+1}\}$ are achieved the weights of the new particles can be calculated as follows:

$$\omega_r = \frac{1}{c}\prod_{j=1}^{M} e^{-\frac{(\phi_{j,t+1}-\hat{\phi}_{j,t+1,r})^2}{2\hat{\sigma}^2_{j,t+1,r}}}$$

where

$\hat{\phi}_{j,t+1,r} = \frac{\hat{\phi}_{j,1,t+1,r}+\hat{\phi}_{j,2,t+1,r}}{2}$, $\hat{\sigma}_{j,t+1,r} = |\frac{\hat{\phi}_{j,2,t+1,r}-\hat{\phi}_{j,1,t+1,r}}{4}| + \sigma_v$ and c is a normalization factor.

The next step is to associate line segments $\Gamma_{j,t+1,r}$ to the update particles. It is clear that for each particle $\mathbf{x}^R_{t,i}$ a predicted distribution can be assumed such as $p(\mathbf{x}^R_{t+1}|\mathbf{x}^R_{t,i})$. However, from this distribution we expect only $n_{i,t+1} = N_p \times p(\mathbf{x}^R_{t,i}|\Phi_{j,t+1})$ samples to exist in $\hat{X}^R_{t+1}$. Therefore, considering Fig. 4.4, in order not to filter out important hypothesizes of a landmark position in sampling phases, $\Gamma_{j,t+1,r}$s should be cut out from $\Gamma_{j,t,i}$ cautiously. A robust way for this purpose is to intersect $\Gamma_{j,t,i}$ with two lines starting at the point $\hat{\mathbf{x}}^R_{t+1,r}|\mathbf{x}^R_{t,i}$ with the following angles:

$\psi_{j,r,1} = \hat{\theta}_{t+1,r,1} + \phi_{j,t+1} + 2\sigma_v + \sigma_{w2}$ and

$\psi_{j,r,2} = \hat{\theta}_{t+1,r,1} + \phi_{j,t+1} - 2\sigma_v - \sigma_{w2}$.

4.2 A Subspace Method with Minimal Number of Samples

The method offered in the previous section was based on a particle filter approach. In particle filter based methods, the sufficient number of particles is always an issue so that the number of particles should be increased exponentially if the the uncertainty of a state vector or the dimension of the vector increase. To alleviate this problem inspired from the FastSLAM2.0 [MTKW03], we use the current measurement to shrink the uncertainties of the robot pose. Consequently, the number of samples can be kept constant in a wide range of odometry noise. To sample the PDF of the robot pose efficiently with a minimal number of samples, we use the sampling method used in the UKF filter. As already mentioned,

the sampled points are known as sigma points. In the UKF sampling method, from a vector of the length n, $2n+1$ samples are obtained. Consequently, given the robot pose vector $[x_t^R, y_t^R, \theta_t^R]^T$ with the distribution $\mathbf{x}_t^R \sim \mathcal{N}(\bar{\mathbf{x}}_t^R, \Sigma)$, seven samples can be obtained:

$$\begin{aligned}
\mathbf{x}_{t,0}^R &= \bar{\mathbf{x}}_t^R \\
\mathbf{x}_{t,r}^R &= \bar{\mathbf{x}}_t^R + \left(\sqrt{\frac{3}{1-w_0}\Sigma}\right)_r \\
\mathbf{x}_{t,r+3}^R &= \bar{\mathbf{x}}_t^R - \left(\sqrt{\frac{3}{1-w_0}\Sigma}\right)_r \\
w_0 &= \frac{1}{3} \\
w_r &= w_{r+3} = \frac{1}{9}; \quad r = 1,2,3
\end{aligned} \tag{4.14}$$

where $(.)_r$ is the r^{th} column of the matrix.

4.2.1 Intersection of Two Line Segments Under the Uncertainties of Parameters

Similar to the particle filter based method, we should localize the landmarks using the intersection of line segments. However, since we use minimal number of samples, the parameters of the line segments have uncertainties and consequently the intersection points have also uncertainties. In this section, we discuss given a sample such as $\mathbf{x}_{t,i}^R$ from $p(\mathbf{x}_t^R|.)$, $u_{f,t}$, $u_{r,t}$ and $\Phi_{j,t+1}$, what would be the distribution of the intersection points.

We know that a line equation can be presented in a vector form as follows:

$$\mathbf{x} = \mathbf{x}_0 + d\mathbf{p} \tag{4.15}$$

where $\mathbf{x} = [x\ y]^T$ represents the point locations on the line, $\mathbf{x}_0 = [x_0\ y_0]^T$ is a known point of the line, d is a scalar depth variable and $\mathbf{p} = [p_1\ p_2]^T$ is a vector which stands for the orientation of the line. In a line equation, the depth

parameter varies between $-\infty$ and $+\infty$. For a line segment, however, it lies in a limited interval such as $[d_{min}, d_{max}]$.

We should mention that in the line segment method once a landmark at the time step l is initialized with a line segment such as $\Gamma_{j,l,t}$, its orientation is kept almost constant but its starting point and the depth parameter will be changed by receiving the new measurements. The changes of these parameters are based on the intersection point of the two lines: $\Gamma_{j,l,k,i}$ and the line at where the same landmark at the new robot pose is observed ($\Gamma_{j,k+1,i}$). For the simplicity, we remove the index i and also replace the indexes (j, l, k) and $(j, k+1)$ with 1 and 2. Therefore, based on Eq. (4.15), the line equations will be:

$$\begin{aligned} \mathbf{x}_1^L &= \mathbf{x}_{1,0}^L + d_1 \mathbf{p}_1 \\ \mathbf{x}_2^L &= \mathbf{x}_{2,0}^L + d_2 \mathbf{p}_2 \end{aligned} \tag{4.16}$$

where $\mathbf{x}_{2,0}^L = [x_{t+1}^R \quad y_{t+1}^R]^T$, $\mathbf{p}_1 = [\cos\alpha_1 \quad \sin\alpha_1]^T$: $\alpha_1 = \theta_l^R + \phi_{j,l}$ and $\mathbf{p}_2 = [\cos\alpha_2 \quad \sin\alpha_2]^T$: $\alpha_2 = \theta_{t+1}^R + \phi_{j,k+1}$.

Eq. (4.16) can be solved for d_1:

$$\begin{bmatrix} \mathbf{p}_1 | -\mathbf{p}_2 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} \Delta x^L \\ \Delta y^L \end{bmatrix} \tag{4.17}$$

where $\Delta x^L = x_{2,0}^L - x_{1,0}^L$, $\Delta y^L = y_{2,0}^L - y_{1,0}^L$.

The solution for d_1 is:

$$d_1 = \frac{\Delta x^L \sin\alpha_2 - \Delta y^L \cos\alpha_2}{\sin(\alpha_2 - \alpha_1)} \tag{4.18}$$

Considering the motion model of a differential drive robot:

$$\begin{aligned} x_{t+1}^R &= x_t^R + u_{f,t}\cos(\theta_t^R) \\ y_{t+1}^R &= y_t^R + u_{f,t}\sin(\theta_t^R) \\ \theta_{t+1}^R &= \theta_t^R + u_{r,t} \end{aligned} \tag{4.19}$$

Eq. (4.18) can be rewritten as:

$$d_1 = \frac{(x_t^R - x_{1,0}^L)\sin\alpha_2 - (y_t^R - y_{1,0}^L)\cos\alpha_2 + u_{f,t}\sin(\alpha_2 - \theta_t^R)}{\sin(\alpha_2 - \alpha_1)} \tag{4.20}$$

Since $\phi_{j,l}$, ϕ_{t+1} and $u_{f,t}$ are Gaussian random variables as follows:

$$\begin{aligned} \phi_{j,l} &\sim \mathcal{N}(\bar{\phi}_{j,l}, \sigma_v^2) \\ \phi_{j,t+1} &\sim \mathcal{N}(\bar{\phi}_{j,t+1}, \sigma_v^2) \\ u_{f,t} &\sim \mathcal{N}(\bar{u}_{f,t}, \sigma_{\tilde{u}_f}^2) \end{aligned} \tag{4.21}$$

we can write:

$$\begin{aligned} \alpha_1 &= \bar{\alpha}_1 + \tilde{\alpha}_1 \\ \alpha_2 &= \bar{\alpha}_2 + \tilde{\alpha}_2 \end{aligned} \tag{4.22}$$

where $\bar{\alpha}_1 = \bar{\theta}_l^R + \bar{\phi}_{j,l}$, $\tilde{\alpha}_1 \sim \mathcal{N}(0, \sigma_v^2)$ and $\bar{\alpha}_2 = \theta_t^R + \bar{u}_{r,t} + \bar{\phi}_{j,t+1}$, $\tilde{\alpha}_2 \sim \mathcal{N}(0, \sigma_v^2 + \sigma_{\tilde{u}_r}^2)$.

By plugging Eq. (4.22) into Eq. (4.20) we have:

$$d_1 = \frac{\begin{array}{c}(x_t^R + u_{f,t}\cos\theta_t^R - x_{1,0}^L)\sin(\bar{\alpha}_2 + \tilde{\alpha}_2) - \\ (y_t^R + u_{f,t}\sin\theta_t^R - y_{1,0}^L)\cos(\bar{\alpha}_2 + \tilde{\alpha}_2)\end{array}}{\sin(\bar{\alpha}_2 - \bar{\alpha}_1 + \tilde{\alpha}_2 - \tilde{\alpha}_1)} \tag{4.23}$$

Simplifying Eq. (4.23) gives:

$$d_1 = \frac{\begin{array}{c}\Delta x \sin(\bar{\alpha}_2 + \tilde{\alpha}_2) - \Delta y \cos(\bar{\alpha}_2 + \tilde{\alpha}_2) + \\ (\bar{u}_{f,t} + \tilde{u}_{f,t})\sin(\bar{\alpha}_2 - \theta_t^R + \tilde{\alpha}_2)\end{array}}{\sin(\Delta\bar{\alpha} + \Delta\tilde{\alpha})} \tag{4.24}$$

where $\Delta x = x_t^R - x_{1,0}^L$, $\Delta y = y_t^R - y_{1,0}^L$, $\Delta\bar{\alpha} = \bar{\alpha}_2 - \bar{\alpha}_1$ and $\Delta\tilde{\alpha} = \tilde{\alpha}_2 - \tilde{\alpha}_1$.

4.2.2 Posterior Distributions of $\tilde{u}_{f,t}$ and $\tilde{\alpha}_2$ Given the Landmark Positions

To minimize the uncertainty of the robot pose at each time step, we use Eq. (4.24) to obtain posterior distributions for $\tilde{u}_{f,t}$ and $\tilde{\alpha}_2$. Solving Eq. (4.18) for $\tilde{u}_{f,t}$ and $\tilde{\alpha}_2$ results in the following equations:

$$\tilde{u}_{f,t} = \frac{\begin{array}{c} d_1 \sin(\Delta\bar{\alpha} + \Delta\tilde{\alpha}) - \Delta x \sin(\bar{\alpha}_2 + \tilde{\alpha}_2) + \\ \Delta y \cos(\bar{\alpha}_2 + \tilde{\alpha}_2) - \bar{u}_f \sin(\bar{\alpha}_2 - \theta_t^R + \tilde{\alpha}_2) \end{array}}{\sin(\bar{\alpha}_2 - \theta_t^R + \tilde{\alpha}_2)} \tag{4.25}$$

$$\tilde{\alpha}_2 = \tan^{-1} \frac{\begin{array}{c} \Delta x \sin\bar{\alpha}_2 - \Delta y \cos\bar{\alpha}_2 + \\ (\bar{u}_{f,t} + \tilde{u}_{f,t}) \sin(\bar{\alpha}_2 - \theta_t^R) - d_1 \sin(\Delta\bar{\alpha} - \tilde{\alpha}_1) \end{array}}{\begin{array}{c} -\Delta x \cos\bar{\alpha}_2 + \Delta y \sin\bar{\alpha}_2 + \\ u_f \cos(\bar{\alpha}_2 - \theta_t^R) + d_1 \cos(\Delta\bar{\alpha} - \tilde{\alpha}_1) \end{array}}$$

Calculation of the posterior distributions of $\tilde{u}_{f,t}$ and $\tilde{u}_{r,t}$ from the above equations are not straight forward. Nevertheless we can calculate 2σ equivalent boundaries for them. Given the 2σ borders of $\tilde{u}_{f,t}$, $\tilde{\alpha}_1$, $\tilde{\alpha}_2$ and a line segment such as $\bar{\mathbf{x}}_{j,l,k,i}^L$, we define the following intervals:

$$\Sigma_{u_f} = [-2\sigma_{\tilde{u}_f}, 2\sigma_{\tilde{u}_f}] \tag{4.26}$$

$$\Sigma_{\alpha_1} = [-2\sigma_{\tilde{\alpha}_1}, 2\sigma_{\tilde{\alpha}_1}] \tag{4.27}$$

$$\Sigma_{\alpha_2} = [-2\sigma_{\tilde{\alpha}_2}, 2\sigma_{\tilde{\alpha}_2}] \tag{4.28}$$

$$\Sigma_{d,j,i} = [0, \lambda_{j,l,i,t}] \tag{4.29}$$

By plugging the eight combinations of the borders of Σ_{α_1}, Σ_{α_2} and $\Sigma_{d,j,i}$ into Eq. (4.25), eight values for $\tilde{u}_{f,t}$ are obtained. Consequently, an interval including the minimum and maximum of the eight values can be formed as follows:

$$R_{\tilde{u}_{f,k,j}} = [\tilde{u}_{f,k,j,min}, \tilde{u}_{f,k,j,max}]$$

For each landmark a similar interval can be obtained. Obviously, the best valid interval can be obtained by the intersection of all these intervals with the prior interval of $\tilde{u}_{f,t}$:

$R_{\tilde{u}_{f,t}} = [\tilde{u}_{f,k,1}, \tilde{u}_{f,k,2}] = \left(\bigcap_{j=1}^{M} R_{\tilde{u}_{f,k,j}}\right) \cap \Sigma_{\tilde{u}_{f,t}}$. Using $R_{\tilde{u}_{f,t}}$, finally we can obtain the following posterior distribution for $\tilde{u}_{f,t}$

$$\tilde{u}^p_{f,t} \sim \mathcal{N}\left(\frac{\tilde{u}_{f,k,1} + \tilde{u}_{f,k,2}}{2}, \frac{(\tilde{u}_{f,k,1} - \tilde{u}_{f,k,2})^2}{4}\right) \tag{4.30}$$

Using the same procedure, a posterior distribution for $\tilde{\alpha}_2$ denoted as $\tilde{\alpha}^p_2$ is obtained. Since $\tilde{\alpha}_2 = \tilde{u}_{r,t} + \tilde{\phi}_{j,t}$ and since $\tilde{\phi}_{j,t}$ is a zero mean random Gaussian variable, the best posterior estimation $\tilde{u}^p_{r,t}$ is equal to $\tilde{\alpha}^p_2$.

To weight the samples, we utilize the Mahalanobis distance between the prior and posterior distributions. We should remind that the obtained posterior distributions are for a given sample. To be more specific, we add the index i to the posterior distributions $\tilde{u}^p_{f,k,i}$ and $\tilde{\alpha}^p_{2,i}$. Consequently, the weight of the sample i will be:

$$w_{i,t+1} = \eta w_{i,t} \exp\left[-\frac{(\bar{u}^p_{f,t,i} - \bar{u}_{f,t})^2}{2\sigma^2_{f,i}} - \frac{(\bar{\alpha}^p_{2,i} - \bar{\alpha}_{2,i})^2}{2\sigma^2_{r,i}}\right] \tag{4.31}$$

where $\sigma^2_{f,i} = \sigma^2_{\tilde{u}^p_{f,t,i}} + \sigma^2_{\tilde{u}_{f,t}}$, $\sigma^2_{r,i} = \sigma^2_{\tilde{\alpha}^p_2} + \sigma^2_{\tilde{\alpha}_2}$ and η is a normalization factor.

In common Bayesian filter methods, the update of a distribution is usually based on the Mahalanobis distance between the predicted measurements and the real measurements. However, in our method, we calculated this relation backward and the updated distribution was obtained based on the comparison between prior and posterior distributions of the input commands.

4.2.3 Optimal Trimming of Line Segments

After obtaining the posterior distributions for $\tilde{u}^p_{f,t}$ and $\tilde{u}^p_{r,t}$, now we should obtain the projection of $p(\mathbf{x}^R_{t+1}|\mathbf{x}^R_t, \tilde{u}^p_{f,t}, \tilde{u}^p_{r,t})$ on each line segment. We drop the upper index p in the following for the sake of simplicity.

d_1 is a random variable which is a function of three random variables $\tilde{\alpha}_1$, $\tilde{\alpha}_2$ and $\tilde{u}_{f,t}$. If $|\tilde{\alpha}_1|, |\Delta\tilde{\alpha}| < 0.15$ rad, the sin(.) function can be approximated using the first order term in its Tailor expansion. Consequently, Eq. (4.18) can be written as follows:

$$d_1 \approx \frac{A}{B + C\Delta\tilde{\alpha}} = \frac{\bar{A} + \tilde{A}}{B + C\Delta\tilde{\alpha}} \tag{4.32}$$

where

$$\begin{aligned}
\bar{A} =& \Delta y \cos\bar{\alpha}_2 - \Delta x \sin\bar{\alpha}_2 + \bar{u}_{f,t}\sin(\bar{\alpha}_2 - \theta_t^R),\\
\tilde{A} =& \left(-\Delta y \sin\bar{\alpha}_2 - \Delta x\cos\bar{\alpha}_2 + \bar{u}_{f,t}\cos(\bar{\alpha}_2 - \theta_t^R)\right)\tilde{\alpha}_2\\
&+ \tilde{u}_{f,t}\sin(\bar{\alpha}_2 - \theta_t^R) = \gamma_1\tilde{\alpha}_1 + \gamma_2\tilde{u}_{f,t},\\
B =& \sin\Delta\bar{\alpha}, \quad C = \cos\Delta\bar{\alpha}
\end{aligned}$$

For larger $\Delta\tilde{\alpha}$, the above approximations will be gradually violated. However, since sin(.) function compresses a zero mean random variable towards zero, using the approximation in the derivation of the equations means that we considered larger uncertainties rather than the real cases which makes our calculation still valid but not optimal.

In Eq. (4.32), A has a Gaussian distribution: $\mathcal{N}(\bar{A}, \gamma_1^2\sigma_{\alpha_1}^2 + \gamma_2^2\sigma_{\tilde{u}_{f,t}}^2)$. If we obtain the distribution of $p(d_1|A)$ then $p(d_1)$ can be calculated as follows:

$$p(d_1) = \int p(d_1|A)p(A)dA \tag{4.33}$$

The above equation means to smooth $p(d_1|A)$ with a Gaussian kernel $\mathcal{N}(0, \gamma_1^2\sigma_{\alpha_1}^2 + \gamma_2^2\sigma_{\tilde{u}_{f,t}}^2)$. The smoothing operation expands $p(d_1|A)$ proportional to the standard deviation of the Gaussian. Therefore, we firstly calculate $p(d_1|\bar{A})$ and with the mentioned consideration, trim the line segment optimally. Using the random variable algebra, the PDF of d_1 given A can be obtained as follows:

$$p(d_1|A) = \frac{1}{\sqrt{2\pi}\sigma_{\Delta\tilde{\alpha}}}\frac{|A|}{Cd_1^2}\exp\left(-\frac{(A - Bd_1)^2}{2C^2\sigma_{\Delta_{\tilde{\alpha}}}^2 d_1^2}\right) \tag{4.34}$$

In Fig. 4.6a, the PDF for $A = 1$, $\sigma_{\Delta_{\tilde{\alpha}}} = 0.1$ and $\Delta\bar{\alpha} = 0$, 0.1 can be seen.

The integral in Eq. (4.33) cannot be solved analytically. However, as we already mentioned, it should look like $p(d_1|\bar{A})$ but smoothed and expanded (Fig. 4.6b). The extrema points of $p(d_1)$ are the key-points to trim a line segment, considering the fact that d_1 should be always non negative. The PDF $p(d_1)$ has three extrema

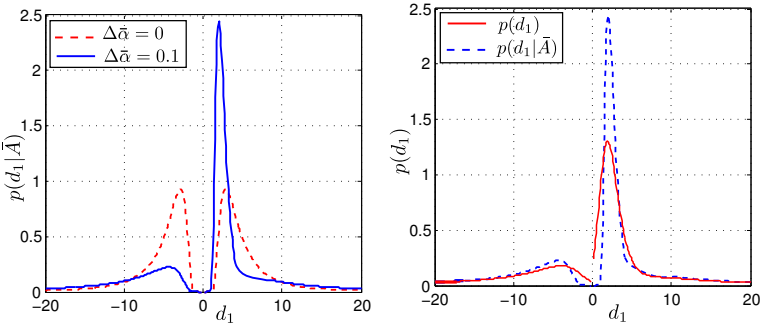


Figure 4.6: (a) $p(d_1|\bar{A})$ for two different parallax angles. (b) $p(d_1|\bar{A})$ and $p(d_1)$ for $\Delta\bar{\alpha} = 0.1$. In both cases, the standard deviation of the random variables are: $\sigma_{\tilde{u}_{f,t}} = \sigma_{\tilde{\alpha}_2} = 0.1$ and $\sigma_{\tilde{\alpha}_1=0.001}$

points if $B \neq 0$. The extrema points can be obtained as follows:

$$\frac{\partial p(d_1)}{\partial d_1} = 0 \tag{4.35}$$

However, since the closed form of $p(d_1)$ is not available, based on the above discussions, We firstly calculate the extrema of $p(d_1|\bar{A})$ and then consider the effect of the uncertainty of $\bar{A}$. Therefore we have:

$$\frac{\partial p(d_1|A)}{\partial d_1} = 2\sigma^2 d_1^2 + ABd_1 - A^2 = 0 \tag{4.36}$$

The solution of this equation always results in real roots one of which is at least positive. By inserting the two roots from Eq. (4.36) in Eq. (4.32) two solutions for $\Delta\tilde{\alpha}$ such as $\Delta_1\tilde{\alpha}$ and $\Delta_2\tilde{\alpha}$ are obtained. We can compensate the effect of $\Delta\tilde{\alpha}$ by recalculation of Eq. (4.32), given $\Delta_r\tilde{\alpha}$, $r = 1, 2$. It results in three distances

such as $d_{1,1} < d_{1,2}$. To compensate the effect of $\tilde{u}_{f,t}$ we use its 2σ confidence region to obtain the following term:

$$b_r = \left| \frac{2\sigma_{\tilde{u}_{f,t}} \sin(\bar{\alpha}_2 - \theta_t^R)}{\sin(\Delta\bar{\alpha} + \Delta_r\tilde{\alpha})} \right|; \quad r = 1, 2 \tag{4.37}$$

Then we modify $d_{1,r}$ as follows:

$$d_{1,j,r} = \begin{cases} d_{1,r} - b_r, & \text{if } d_{1,r} \geq 0 \\ d_{1,r} + b_r, & \text{if } d_{1,r} < 0 \end{cases} \tag{4.38}$$

We sort the obtained values ascending: $d_{1,1} < d_{1,2}$ and define the following interval based on the two greatest values:

$$\bar{d_1} = [d_{1,2}, 2\frac{d_{1,3} - d_{1,2}}{u(|\Delta\alpha| - 2\sigma_{\Delta\tilde{\alpha}})}] \tag{4.39}$$

where $u(.)$ is a step function. After the calculation of the above interval, $\bar{d_1}$ is intersected with the interval $[0, \lambda_{j,l,i}]$ to obtain the updated line segment. In Eq. (4.39), the infinity depth is preserved if the parallax angle is less than the confidence region of $\Delta\tilde{\alpha}$.

4.2.4 Resampling

We can fit a Gaussian distribution to the statistics including seven weighted samples. To obtain a Gaussian distribution for the robot pose based on the statistics and also preserve the dependency between the robot and landmark positions, we define the following vector for each set of robot sample and the parameters of the line segment:

$$\mathbf{x}_{i,j} = [x_i, y_i, \theta_i, x_{j,i}^L, y_{j,i}^L, \alpha_{j,i}^L, \lambda_{j,i}^L]^T \tag{4.40}$$

Consequently, the following mean vector and covariance matrix can be obtained:

$$\begin{aligned} \mathbf{x}_j &= \sum_{i=1}^{7} w_i \mathbf{x}_{i,j} \\ \Sigma_j &= \sum_{i=1}^{7} w_i (\mathbf{x}_{i,j} - \mathbf{x}_j)(\mathbf{x}_{i,j} - \mathbf{x}_j)^T \end{aligned} \tag{4.41}$$

Then the new samples are obtained as follows:

$$\begin{aligned} \mathbf{x}_{0,j} &= \mathbf{x}_j \\ \mathbf{x}_{r,j} &= \mathbf{x}_j - \left(\sqrt{\frac{9}{2}\Sigma_j}\right)_r \\ \mathbf{x}_{3+r,j} &= \mathbf{x}_j + \left(\sqrt{\frac{9}{2}\Sigma_j}\right)_r \end{aligned}$$

where $r = 1, 2, 3$. Obviously, for all robot and landmark combinations the mean and covariance of the robot pose will be the same but we have to calculate the mean vector and covariance matrices the way mentioned above, in order not to lose the dependency between the robot pose and the landmark positions.

4.3 Complexity Analysis

Unlike the EKF based methods, the complexity of the proposed algorithms grow linearly with respect to the number of landmarks (M). The complexity of EKF based methods have direct relation to the size of their covariance matrices. Hence, the IDP (inverse depth parametrization) and LDP (logarithmic depth parameterization) methods in 2D cases have the complexity of $O\big((3+4M)^2\big)$ since four parameters are used to initialize each landmark in the state vector. For the GSF based method, since it uses n_g^M parallel EKF filters (n_g is the number of Gaussians used to model each landmark uncertainty), its complexity is $O\big(n_g^M(3+2M)^2\big)$.

The complexity of our particle filter based algorithm is $O\big(2MN\big)$ where N is the number of particles. The complexity is related to the calculation of intersected

lines at each step. It means if e.g. 150 particles are used, our algorithm has less complexity than the inverse depth parametrization method for the number of landmarks more than eighteen.

Concerning the subspace method algorithm, its complexity also grows linearly with respect to the number of landmarks (M) which is $O\big(N^2 \times M\big)$ where N is the number of samples. This complexity is related to the resampling phase of the algorithm. For $N = 7$, we can conclude that for any number of landmarks our algorithm has less complexity than the inverse depth parametrization method.

4.4 Simulation

To show the performance of the proposed algorithms, an environment including one robot and 80 landmarks was simulated. The robot was driven firstly on a linear path which is the most challenging case for monocular SLAM since many landmarks are observed at low parallax angles for many steps. Then to observe the performance of the algorithms in the case of rotations, we let the robot also rotate on a circular path as well. For the linear part, the forward and rotation speeds were $u_f = 0.3$ m/step, $u_r = 0$ rad/step and for the circular part, they were $u_f = 0.3$ m/step, $u_r = 0.08$ rad/step. The standard deviation of odometry noise varied between 0.02 to 0.1 (m/step or rad/step). We compared our methods with IDP (inverse depth parametrization), IIDP (iterative inverse depth parametrization), LDP (logarithmic depth parametrization), GSF (Gaussian sum filter) and a SLAM method based on UKF. We notify that the computation time of GSF method for more than 5 landmarks is getting extremely high and for more than 10 landmarks running the algorithm is not feasible. Hence, we used 5 randomly selected landmarks out of 80 ones and ran the GSF method. To minimize the effect of random selection of landmarks, we ran the simulation several times and then calculated estimation errors in average. The root mean square error (RMSE) between the ground truth and the real paths are shown in Fig. 4.7. Additionally, to achieve a visual perception of the performances of different methods, the estimated paths and the landmark positions based on different methods for $\sigma_{u_f} = \sigma_{u_r} = 0.05$ are depicted in Fig. 4.8.

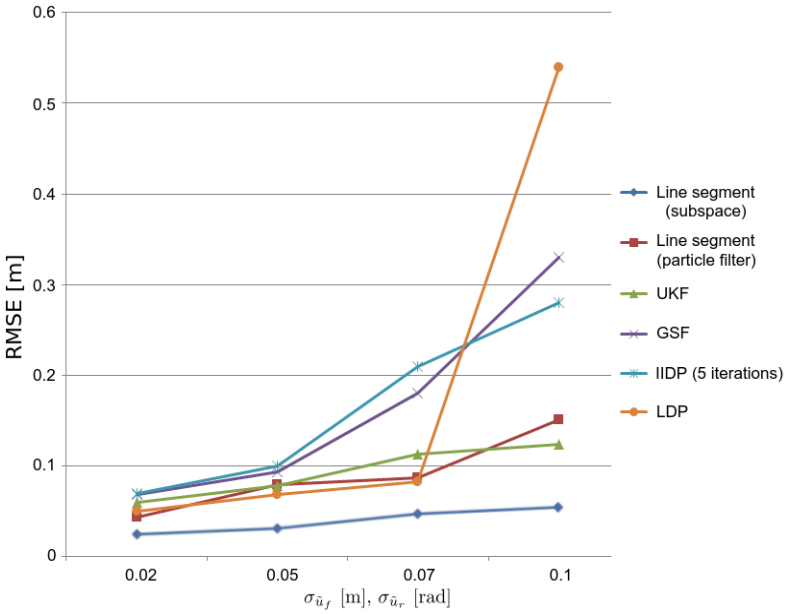
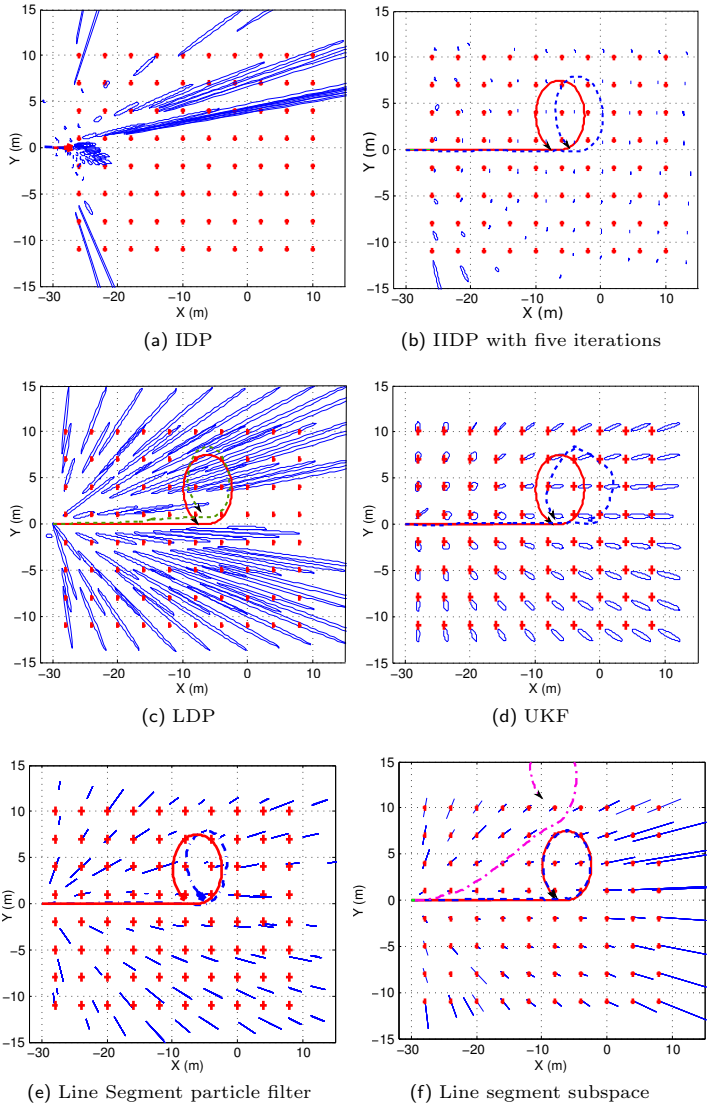


Figure 4.7: Root mean square error between the ground truth and estimated paths for different methods.

We can observe that the IDP method only managed to converge with five iterations in its update phase. But even in this condition it had a biased estimation of the robot and landmark positions. The LDP method had a better estimation of the path. However, it could not reduce the landmark uncertainties effectively and also it started to diverge for the odometry noises with the standard deviations more than 0.05. The reason is the usage of second order Kalman filters which include high amount of uncertainties even if the landmarks are observed at high parallax angles. Concerning the line segment based methods, we see that the particle filter based method with 50 particles gave results better than the IIDP method (with much lower complexity). Finally, for the subspace method, we see that the path was estimated very well for the wide range of the odometry noises.



(a) IDP (b) IIDP with five iterations

(c) LDP (d) UKF

(e) Line Segment particle filter (f) Line segment subspace

Figure 4.8: Localization and mapping using different methods for $\sigma_{u_f} = 0.5$ m/step, $\sigma_{u_r} = 0.05$ rad/step and $\sigma_v = .001$ rad. The continuous paths (red) are the ground truth paths and the dashed paths (blue) are the estimated paths using different methods. The estimated position of landmarks are shown with ellipses or line segments. The pink path in (f) is the odometry path.

An important issue concerning the UKF based method is that it highly depends on the initial depths for the landmarks. It should be almost near to the average depths of the landmarks, otherwise the algorithm diverges. Fig. 4.9 shows the RMSE for three different initial depth values. We see that it only converged at the initial depth values of 30 m and for greater or smaller initial values, the UKF method has diverged. It means that for this method, we still need some prior knowledge about the uncertainties of the depths in an environment.

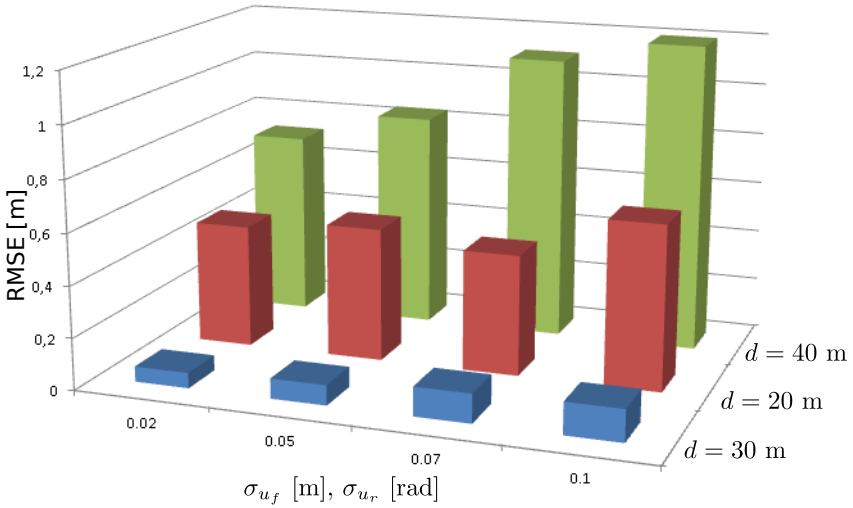


Figure 4.9: Root Mean square error between the ground truth and estimated paths based on the UKF filter depending on the initial depths for landmarks.

4.5 Experimental Results

We implemented the line segment algorithms on a pioneer four wheeled robot which was equipped with a laptop with an Intel Core 2 Duo (3.33 GHz) processor. A consumer webcam with a resolution of 960×720 and the field of view about 60° was also installed on the robot. We ran two experiments: an outdoor and

an indoor scenario. For both cases we used the forward backward Lucas-Kanade tracker to track Harris corner features.

In our application, landmarks were extracted from a frame and tracked in the next frames until 70% of them disappeared. Obviously, since the algorithms are developed for 2D case, we only used the displacement of the features along the horizontal vector of the camera to achieve the bearing measurements.

To detect outliers, the predicted ranges of measurements were taken into account. The ranges were obtained based on the projections of landmarks on the retina of the camera, odometry data and the minimum and the maximum expected depths of the landmarks.

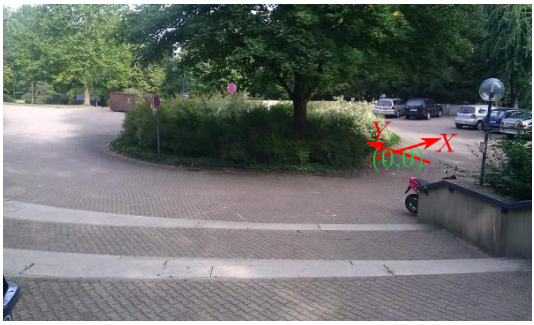


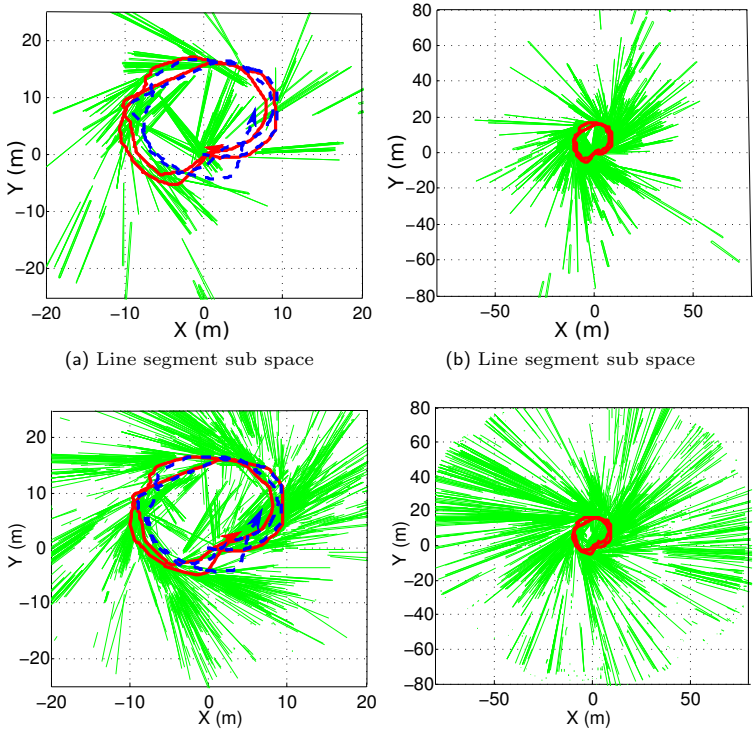
Figure 4.10: The square about which the robot was driven. The starting point and its $X - Y$ coordinate are marked.

The initial number of landmarks was 150 by which the algorithms managed to work in real time at the frame rate 20 Hz This rate was used if the robot rotated and for the linear movements, we only used 5 Hz. For the particle filter based method the number of particles was set to 100.

For the outdoor scenario, the robot was driven two times about an square. The square and the starting point of the run can be seen in Fig. 4.10. The robot passed the starting point two times. The generated path by the odometry and the modified path using the algorithm in addition to the extracted landmarks with the length uncertainties less than 15 m can be seen in Fig. 4.11a and Fig.

4.11c. It can be observed that the subspace method had a high performance such that it managed to correct a relatively high amount of odometry errors and close the loop with a precision 0.56 m. The particle filter based method also had a good performance with the error 1.23 m in both turns. Our method extracted and registered about 2000 landmarks. The landmark uncertainties are presented as line segments. It should be mentioned that although the life times of the tracks of the landmarks were mostly short (in average less than ten frames), and also in outdoor scenarios we should consider relatively high measurement noises due to the vibration of the camera, the proposed algorithms managed to extract many landmarks with the length uncertainties less than 15 m.

The second experiment was conducted in an indoor hallway. The run included both straight and rotational motions. The map of the hallway, the odometry and modified paths and additionally the landmarks with the length uncertainties less than 1 m can be seen in Fig. 4.12. We can observe that the subspace method managed again to correct the odometry path very well and also localized lots of landmarks with low uncertainties. The particle filter based method, however, had an error of about 3 m. The indoor experience was challenging since there were not many robust features to track and also due to the shadowing, there were lots of outliers which could be hardly detected.



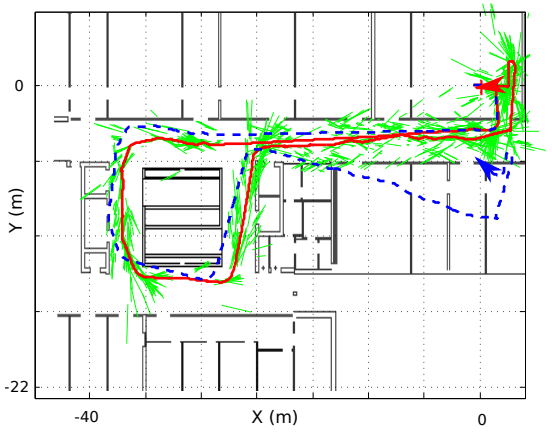
(a) Line segment sub space

(b) Line segment sub space

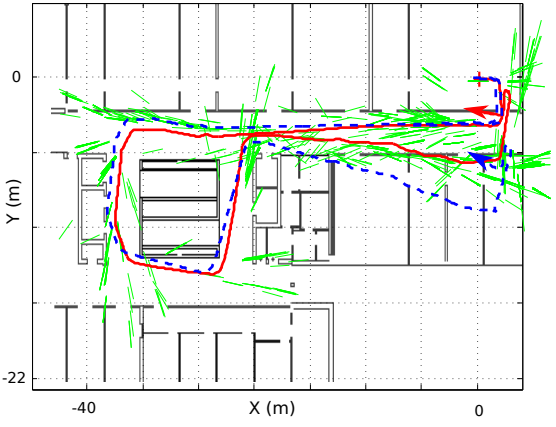
(c) Line segment particle filter

(d) Line segment particle filter

Figure 4.11: Outdoor experiment: Odometry path (dashed curve), modified paths generated by the line segment methods (continuous curve). (a),(c) Extracted landmarks with length uncertainties less than 15 m. (b),(d) Extracted landmarks with the length uncertainties less than 60 m.



(a) Line segment sub space



(b) Line segment particle filter

Figure 4.12: Indoor experiment: Odometry path (dashed curve), modified paths generated by line segment methods (continuous curve) and extracted landmarks with length uncertainties less than 1.5 m.

5 Conclusion and Outlook

In this chapter, the contribution of this work to monocular visual odometry and SLAM problems is summarized and future work is discussed.

5.1 Conclusion

In this thesis, two related fields of monocular SLAM has been investigated: visual odometry and data fusion. Concerning visual odometry, we studied feature tracking in consecutive frames using two approaches: (i) the feature matching approach based on the SIFT, SURF, ORB and BRISK descriptors and (ii) the sparse optical flow method (Lucas-Kanade). Conventionally, feature trackers are evaluated based on only a limited set of paired images captured from planar surfaces. The reason for using planar surfaces is that ground truth for matched features can accurately be calculated using homography transforms. We used long sequences from the training KITTI dataset for visual odometry to evaluate different trackers. In this regard, due to the availability of ground truth for camera poses in the dataset, we were able to provide appropriate measures using epipolar geometry to evaluate different methods in highly challenging outdoor environments. We also discussed how to track features in several consecutive frames robustly. Different feature trackers were evaluated based on four measures: endurance, precision, recall and measurement noise. On the basis of these measures, it was demonstrated that the forward backward Lucas-Kanade method outperformed the trackers using the feature matching approach to a great extent. Actually, use of the Lucas-Kanade method without the forward-backward manner typically results in a high percentage of outliers. It could be the reason that in most of the relevant literature feature matching techniques such as SIFT and SURF have been used. Interestingly, applying the forward-backward

technique, many of the outliers can be detected and removed. We also investigated the poor performance of different N-point methods for the relative camera motion recovery. The major problem stemmed from ignoring the effect of noise in each coplanarity equation. It was shown that in the presence of measurement noise, each coplanarity equation can deviate from zero. By considering relatively large uncertainties for motion parameters, we obtained deviations for each equation and solved the related homogeneous equation system based on the Mahalanobis distance. Additionally, a regularization term in the form of a covariance matrix for the essential matrix elements was obtained which regularized the final solution based on the physical constraints of cameras. These two modifications yielded significant improvements for the 8-point and 7-point methods, which was proved by experiments with simulated data and the KITTI dataset. We also introduced a modified Cheirality test which was simpler and more effective than previous Cheirality test especially for the case that a camera moves in depth and observes features at low parallax angles. In this case, the points can be back projected behind the camera wrongly resulting in wrong decisions for motion directions. Furthermore, a new technique was proposed to estimate scale of translations by tracking low quality features on ground planes. As the Lucas-Kanade method can track high quality features (features with relatively high brightness gradients), this method fails to track low quality features on ground planes very often. Hence, the main challenge of scale detection is how to track low quality features. We proposed a new compact descriptor to find for hundreds of low quality features in one image at least two matches in the consecutive image. Many of the wrong matches were filtered out based on the distances of matching candidates to their corresponding epipolar lines. This technique resulted in substantial improvements in the estimation of scale factors. The overall performance of the discussed method was demonstrated based on the training and test KITTI dataset.

We also contributed to the monocular SLAM problem by proposing non-Gaussian techniques for the fusion part of monocular SLAM. The main part of the methods was modeling landmark uncertainties over line segments with a uniform distribution if landmarks are observed at low parallax angles. In these methods, the distribution of a robot pose is presented with weighted samples and attached

to each sample a line segment for each landmark is initialized. Two different methods were proposed to update the samples and to reduce the uncertainties of landmarks (lengths of line segments). In the first method, a particle filter based method was used, in which the distribution of the robot pose was modeled by more than hundred weighted particles and the lengths of the corresponding line segments were reduced by intersecting the lines with trapezoids established from the new observations of landmarks. In this method, the number of particles should be increased if the uncertainties of odometry data increase. To alleviate this problem, a new method was proposed, which used minimal efficient samples of the robot pose distribution that was acceptably assumed Gaussian. Using such a minimal number of samples led to introducing an efficient method for line segment trimming namely probabilistic triangulation. We demonstrated that the depth parameter for a landmark has the distribution of the ratio of two Gaussian random variables. This distribution has a very complex form and extraction of useful information from it is very difficult. To deal with this problem, we used a conditional probability approach to obtain initial guesses for the extrema points of the distribution and then proposed an optimal solution to trim the lengths of line segments. Additionally, to localize robots, instead of iterative optimization techniques, we used a subspace method. If landmarks are observed at low parallax angles, optimization techniques give rise to biased estimation of robot poses and landmark positions in favor of initial beliefs and do not delay decisions concerning robot poses until the time that enough evidences are gathered. Through simulation, we observed this effect using the iterative inverse depth parameterization (IIDP) method which had a poor performance in the estimation of scale of motions and gave rise to a shifted estimation of robot and landmark positions. Finally, we proved the high performance of the line segment and subspace method through simulation and experimental results and showed that the method outperformed other methods to a great extent with a constant computation cost with respect to the amount of odometry uncertainties.

5.2 Outlook

In this thesis, robust techniques for monocular visual odometry and 2D SLAM problems in case of static scenes were proposed. We are extending our proposed line segment technique for the 3D monocular SLAM problem [MM15] and planning to consider highly dynamic scenes, in which several moving objects exist. In a simple case where static background still constitutes most of a scene, the discussed methods could be applied by the detection of the landmarks on moving objects as outliers. On the contrary, if moving objects are dominant, the SLAM problem should be formulated in a different way that the relative velocity of the robot to each object should be taken into account and instead of a static map a dynamic map should be defined in which for each object in addition to its position, its velocity is also considered. Other future work is simultaneous estimation of optical flows and vehicles' motions. In this regard, epipolar geometry can be used to add an extra constraint for the calculation of optical flows resulting in better estimations of flows, especially for low textured scenes [MMM15]. Consequently, the enhanced estimations of flows can be utilized to achieve better estimation of a vehicle's motion.

Bibliography

[AKB08] Agrawal, M.; Konolige, K.; Blas, M. R.: Censure: Center Surround Extremas for Realtime Feature Detection and Matching. In: *European Conference on Computer Vision (ECCV)*, Vol. 5305, 2008, pp. 102–115

[Ana89] Anandan, P.: A Computational Framework and an Algorithm for the Measurement of Visual Motion. In: *International Journal of Computer Vision*, Vol. 2, 1989, No. 3, pp. 283–310

[BA83] Burt, P. J.; ; Adelson, E. H.: The Laplacian Pyramid as a Compact Image Code. In: *IEEE Transactions on Communications*, Vol. 31, 1983, pp. 532–540

[Bai03] Bailey, T.: Constrained Initialisation for Bearing-only SLAM. In: *Proceeding of IEEE International Conference on Robotics and Automation (ICRA)*, Vol. 2, 2003, pp. 1966–1971

[Bur81] Burt, P.J.: Fast Filter Transforms for Image Processing. In: *Computer Graphics and Image Processing*, Vol. 16, 1981, No. 1, pp. 20–51

[CDM08] Civera, J.; Davison, A. J.; Montiel, J. M.: Inverse Depth Parametrization for Monocular SLAM. In: *IEEE Transactions on Robotics*, Vol. 24, 2008, No. 5, pp. 932–945

[CLSF10] Calonder, M.; Lepetit, V.; Strecha, C.; Fua, P.: BRIEF: Binary Robust Independent Elementary Features. In: *Proceedings of the 11th European Conference on Computer Vision*, 2010, pp. 778–792

[CR96] Casella, G.; Robert, C. P.: Rao-Blackwellization of Sampling Schemes. , Vol. 83, 1996, pp. 81–94

[DK06] Dellaert, F.; Kaess, M.: Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. In: *International Journal on Robotics Research*, Vol. 25, 2006, No. 12, pp. 1181–1203

[DNC+01] Dissanayake, G.; Newman, P.; Clark, S.; Durrant-whyte, H. F.; Csorba, M.: A Solution to the Simultaneous Localization and Map building (SLAM) Problem. In: *IEEE Transactions on Robotics and Automation*, 2001, pp. 229–241

[Fau93] Faugeras, O.: *Three-dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993

[FB81] Fischler, M. A.; Bolles, R. C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In: *Communications of the ACM*, Vol. 24, 1981, No. 6, pp. 381–395

[FBDT99] Fox, D.; Burgard, W.; Dellaert, F.; Thrun, S.: Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In: *Proceedings of National Conference on Artificial Intelligence (AAAI)*, 1999, pp. 343–349

[FJ90] Fleet, D. J.; Jepson, A. D.: Computation of Component Image velocity from Local Phase Information. In: *International Journal of Computer Vision*, 1990, pp. 77–104

[GN01] Guivant, J.; Nebot, E.: Optimization of the Simultaneous Localization and Map Building Algorithm for Real Time Implementation. In: *IEEE Transactions on Robotics and Automation*, Vol. 17, 2001, pp. 242–257

[GSS93] Gordon, N.J.; Salmond, D.J.; Smith, A.F.M.: Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation. In: *Proceedings of Radar and Signal Processing*, 1993, pp. 107–113

[GZS11] Geiger, A.; Ziegler, J.; Stiller, C.: StereoScan: Dense 3d Reconstruction in Real-time. In: *Proceeding of Intelligent Vehicles Symposium*, 2011

[Har97] Hartley, R. I.: In Defense of the Eight-Point Algorithm. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Vol. 19, 1997, No. 6, pp. 580–593

[Hee87] Heeger, D. J.: Model for the Extraction of Image Flow. In: *Journal of Optical Society of America*, Vol. 4, 1987, No. 8, pp. 1455–1471

[HS81] Horn, B. K. P.; Schunck, B. G.: Determining Optical Flow. In: *Artificial Intelligence*, Vol. 17, 1981, pp. 185–203

[HS88] Harris, C.; Stephens, M.: A Combined Corner and Edge Detector. In: *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151

[HTG08] Herbert, B.; Tinne, T.; Gool, L. V.: Speeded Up Robust Features (SURF). , Vol. 110, 2008, pp. 346–359

[HZ04] Hartley, R. I.; Zisserman, A.: *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2004

[JU04] Julier, S. J.; Uhlmann, K.: Unscented Filtering and Nonlinear Estimation. In: *Proceeding of IEEE Symposium on Adaptive Systems for Signal Processing, Communication and Control*, 2004, pp. 401–422

[Kal60] Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems. In: *Journal of Basic Engineering*, Vol. 82, 1960, No. 1, pp. 35–45

[KD04] Kwok, N. M.; Dissanayake, G.: An Efficient Multiple Hypothesis Filter for Bearing-Only SLAM. In: *Proceeding of International Conference on Intelligent Robots and Systems*, 2004, pp. 736–741

[KHH+05] Kwok, N. M.; Ha, Q. P.; Huang, S.; Dissanayake, G.; Fang, G.: Bearing-only SLAM Using a SPRT based Gaussian Sum Filter. In:

Proceedings of the IEEE International Conference on Robotics and Automation, 2005, pp. 1109–1114

[KIT15] *KITTI Visual Odometry Dataset.* `http://www.cvlibs.net/datasets/kitti/eval_odometry.php`. Version: 2015. – Accessed: 2015-07-15

[KM07] Klein, Georg; Murray, David: Parallel Tracking and Mapping for Small AR Workspaces. In: *Proceeding of 6th International Symposium on Mixed and Augmented Reality (ISMAR)*. Nara, Japan, November 2007

[Koe84] Koenderink, J. J.: The Structure of Images. In: *Biological Cybernetics*, Vol. 50, 1984, No. 5, pp. 363–370

[LCS11] Leutenegger, S.; Chli, M.; Siegwart, R.: BRISK: Binary Robust invariant scalable keypoints. In: *Proceeding of International Conference on Computer Vision*, 2011, pp. 2548–2555

[LDFP93] Luong, Q.; Deriche, R.; Faugeras, O.; Papadopoulo, T.: *On Determining The Fundamental Matrix: Analysis Of Different Methods and Experimental Results.* 1993

[LH81] Longuet-Higgins, H. C.: A Computer Algorithm for Reconstructing a Scene from Two Projections. In: *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms.* 1981, pp. 61–62

[LH06] Li, H.; Hartley, R. I.: Five-Point Motion Estimation Made Easy. In: *Proceedings of the 18th International Conference on Pattern Recognition*, 2006, pp. 630–633

[LJF00] Leonard, J. J.; Jacob, H.; Feder, S.: A Computationally Efficient Method for Large-scale Concurrent Mapping and Localization. In: *Proceedings of the Ninth International Symposium on Robotics Research*, 2000, pp. 169–176

[LK81] Lucas, B. D.; Kanade, T.: An Iterative Image Registration Technique with an Application to Stereo Vision. In: *Proceeding of 7th*

International Joint Conference on Artificial Intelligence, 1981, pp. 674–679

[Low04] Lowe, D. G.: Distinctive Image Features from Scale-Invariant Keypoints. In: *International Journal of Computer Vision*, Vol. 60, 2004, pp. 91–110

[MAT14] Mur-Artal, R.; Tardos, J.D.: ORB-SLAM: Tracking and Mapping Recognizable Features. In: *Proceeding of Robotics: Science and Systems (RSS) Workshop on Multi View Geometry in Robotics*, 2014

[MM12a] Miksik, Ondrej; Mikolajczyk, Krystian: Evaluation of Local Detectors and Descriptors for Fast Feature Matching. In: *ICPR*, 2012, pp. 2681–2684

[MM12b] Mirabdollah, M. H.; Mertsching, B.: Bearing Only SLAM: A New Particle Filter Based Approach. In: *Proceeding of Conference on Artificial Intelligent System*, 2012, pp. 116–125

[MM12c] Mirabdollah, M. H.; Mertsching, B.: Monocular SLAM: Using Trapezoids to Model Landmark Uncertainties. In: *Proceeding of IEEE Inernational Conference on Robotics and Biomimetics*, 2012, pp. 482–488

[MM13a] Mirabdollah, M. H.; Mertsching, B.: A New Non-Gaussian Filtering Approach to Monocular SLAM Problem. In: *Proceeding of IEEE International Conference on Intelligent Robotics and Applications*, 2013, pp. 550–561

[MM13b] Mirabdollah, M. H.; Mertsching, B.: Single Camera Motion Estimation: Modification of the 8-Point Method. In: *Proceeding of International Conference on Intelligent Robotics and Applications*, 2013, pp. 117–128

[MM14] Mirabdollah, M. H.; Mertsching, B.: On the Second Order Statistics of Essential Matrix Elements . In: *Proceeding of 36th*

German Conference on Pattern Recognition (GCPR 2014), 2014, pp. 547–557

[MM15] Mirabdollah, M. H.; Mertsching, B.: Fast Techniques for Monocular Visual Odometry. In: *Accepted by 37th German Conference on Pattern Recognition (GCPR)*, 2015

[MMM15] M., M. A.; Mirabdollah, M. H.; Mertsching, B.: Differential Optical Flow Estimation Under Monocular Epipolar Line Constraint. In: *Proceeding of 10th International Conference (ICVS)*, 2015, pp. 354–363

[MTKW02] Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B.: FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In: *Proceedings of National Conference on Artificial Itelligence*, 2002, pp. 593–598

[MTKW03] Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B.: FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2003, pp. 1151–1156

[Mur00] Murphy, K.: Bayesian Map Learning in Dynamic Environments. In: *Proceeding of Conference on Neural Information Processing Systems*, 2000, pp. 1015–1021

[Nis04] Nistér, D.: An Efficient Solution to the Five-Point Relative Pose Problem. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Vol. 26, 2004, No. 6, pp. 756–777

[ope11] *OpenCV feature descriptor comparison report.* `http://computer-vision-talks.com/articles/2011-08-19-feature-descriptor-comparison-report/`, 2011. – Accessed: 2015-04-28

[Ope15] *Open Computer Vision Library (OpenCV).* `http://opencv.org/`. Version: 2015. – Accessed: 2015-04-07

[pio15] *Pioneer 3dX Differential Drive Mobile Robot.* `http://www.mobilerobots.com/researchRobots/PioneerP3DX.aspx`. Version: 2015. – Accessed: 2015-04-07

[PJ08] Parsley, M.P.; Julier, S.J.: Avoiding Negative Depth in Inverse Depth Bearing-only SLAM. In: *Proceeding of IEEE International Conference on Intelligent Robots and Systems*, 2008, pp. 2066–2071

[RD05] Rosten, E.; Drummond, T.: Fusing Points and Lines for High Performance Tracking. In: *Proceeding of International Conference on Compuer Vision (ICCV)*, Springer, 2005, pp. 1508–1515

[RRKB11] Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. R.: ORB: An efficient alternative to SIFT or SURF. In: *Proceeding of International Conference on Computer Vision*, 2011, pp. 2564–2571

[SC14] Song, S.; Chandraker, M.: Robust Scale Estimation in Real-Time Monocular SFM for Autonomous Driving. In: *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014

[SCG13] Song, S.; Chandraker, M.; Guest, C.C.: Parallel, Real-time Monocular Visual Odometry. In: *Proceeding of International Conference on Robotics and Automation*, 2013, pp. 4698–4705

[Sin90] Singh, A.: An Estimation-Theoretic Framework for Image-Flow Computation. In: *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 1990, pp. 168–177

[SMD10] Strasdat, H.; Montiel, J. M. M.; Davison, A.: Scale Drift-Aware Large Scale Monocular SLAM. In: *Proceedings of Robotics: Science and Systems*. Zaragoza, Spain, June 2010

[SMDL05] Sola, J.; Monin, A.; Devy, M.; Lemaire, T.: Undelayed Initialization in Bearing only SLAM. In: *Proceeding of IEEE International Conference on Intelligent Robots and Systems*, 2005, pp. 2499–2504

[SSC90] Smith, R.; Self, M.; Cheeseman, P.: Autonomous Robot Vehicles. Springer-Verlag New York, Inc., 1990, Chapter Estimating Uncertain Spatial Relationships in Robotics, pp. 167–193

[TBF+98] Thrun, S.; Burgard, W.; Fox, D.; Hexmoor, H.; Mataric, M.: A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. In: *Proceeding of Conference on Machine Learning*, 1998, pp. 29–53

[TH84] Tsai, R.; Huang, T. S.: Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Vol. 6, 1984, No. 1, pp. 13–27

[TM05] Thrun, S.; Montemerlo, M.: The GraphSLAM Algorithm With Applications to Large-Scale Mapping of Urban Structures. In: *International Journal on Robotics Research*, Vol. 25, 2005, No. 5/6, pp. 403–430

[TMKC08] Tully, S.; Moon, H.; Kantor, G.; Choset, H.: Iterated Filters for Bearing-only SLAM. In: *Proceeding of IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 1442–1448

[Tsa87] Tsai, R.: A Versatile Camera Calibration Technique for High-accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses. In: *IEEE Journal of Robotics and Automation*, Vol. 3, 1987, No. 4, pp. 323–344

[UGVT88] Uras, S.; Girosi, F.; Verri, A.; Torre, V.: A Computational Approach to Motion Perception. In: *Biological Cybernetics*, 1988, pp. 79–87

[WDD02] Williams, S. B.; Dissanayake, G.; Durrant, H.: An Efficient Approach to the Simultaneous Localisation and Mapping Problem. In: *Proceeding of IEEE International Conference on Robotics and Automation (ICRA)*, 2002, pp. 406–411

[Wit83] Witkin, A. P.: Scale-Space Filtering. In: *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, 1983, pp. 1019–1022

[ZHYD11] Zhao, L.; Huang, S.; Yan, L.; Dissanayake, G.: Parallax Angle Parametrization for Monocular SLAM. In: *Proceeding of IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3117–3124

[ZS15] Zhang, Ji; Singh, Sanjiv: Visual-Lidar Odometry and Mapping: Low-Drift, Robust, and Fast. In: *Proceeding of IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2174–2181

List of Notations

Notation	Explanation
(x, y)	Coordinate of a point in the retina of a calibrated camera
(x^u, y^u)	Coordinate of a point on the retina of an uncalibrated camera
ξ	Information vector
c	Class
$\mathbf{d}$	Feature descriptor vector
$\mathbf{p} = [p_x \ p_y \ p_z]^T$	Coordinate of a point in 3D space
Γ	Line segment
$\mathbf{q}$	Quaternion
$\mathbf{u}_t = [u_{f,t} \ u_{r,t}{}_T]$	Motion velocities
$\mathbf{x}_j^L = [x_j^L \ y_j^L]^T$	j^{th} Landmark position
$\mathbf{x}_t^R = [x_t^R \ y_t^R \ \theta_t^R]^T$	Robot pose at time t
$\mathbf{x}_t$	State vector
$\mathbf{z}_t$	Measurement vector
$\mathcal{E}(.)$	Expectation operator
$\mathcal{N}(\mu, \Sigma)$	Gaussian vector distribution with the mean μ and the covariance Σ
$\mathcal{R}$	Real number set
Ω	Information matrix
σ	Standard deviation
E	Essential matrix
F	Fundamental matrix
$f(.)$	a one dimensional function

Notation	Explanation
$F(x, \sigma)$	Convolution of function $f(x)$ with a Gaussian function with the standard deviation σ
H	Hessian matrix
$\mathbf{h}(\mathbf{x})$	Measurement function given the state vector $\mathbf{x}$
$\mathbf{h}_j(\mathbf{x})$	Measurement function for the j^{th} landmark
$\mathbf{h}_j(\mathbf{x}_t^R, \mathbf{x}_j^L)$	Measurement function given the robot pose $\mathbf{x}_t^R$ and the landmakr $\mathbf{x}_j^L$
id	identification number of a feature
$I(x, y)$	Integral image
$I_\alpha(x, y)$	Slanted integral image
$J_\alpha(x, y)$	Jacobian matrix
K	Camera calibration matrix
k	an integer factor
l	level of pyramid
$L(x, y)$	Gray scale image
M	Number of landmarks
M_l	half width of a Kernel at the level l
N_g	Number of Gaussians per landmark
N_p	Number of particles
N_r	Number of robot poses
P	Covariance matrix of a state vector
$p(.)$	probability distribution function
R	Covariance matrix of measurement noises
R	Rotation matrix
s	Variance
U_{t-1}	Set of all motion commands until time
$w(.)$	a one dimensional kernel
X, Y and Z	Axis of a Cartesian coordinate system
Z_t	Set of all measurements until time

List of Abbreviations

Abbreviation	Explanation
BRIEF	Binary Robust Independent Elementary Features
BRISK	Binary Robust Invariant Scale Keypoints
CenSurE	Centered Surround Extrema
DAG	Distributed Averages of Gradients
EKF	Extended Kalman Filter
EM	Expectation Maximization
E. T.	Elapsed Time
FAST	Features from Accelerated Segment Test
FBLK	Forward Backward Lucas-Kanade
GSF	Gaussian Sum Filter
IDP	Inverse Depth Parametrization
IIDP	Iterative Inverse Depth Parametrization
IMU	inertial Measurement Unit
LDP	Logarithmic Depth Parametrization
LK	Lukas-Kanade
LoG	Laplace of Gaussian
MLM-SFM	Monocular Multicore Large Scale SFM
MME_c	Mean of Magnitudes of Errors for estimation of camera positions
MME_a	Mean of Magnitudes of Errors for estimation of camera orientations
PDF	Probability Density Function
ORB	Oriented Robust BRIEF
RANSAC	Random Sample Consensus
RCMPE+GP	Relative Camera Pose Estimation
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization And Mapping
SURF	Speeded Up Robust Features
SVD	Singular Value Decomposition
UKF	Unscented Kalman Filter
V-LOAM	Vision-Lidar Odometry and Mapping
vSLAM	visual SLAM

List of Tables

List of Figures