UNIVERSITÀ DEGLI STUDI DI PADOVA

UNIVERSITA' DEGLI STUDI DI PADOVA

Dipartimento di Ingegneria Industriale DII

Corso di Laurea in Ingegneria Aerospaziale

AN EXPERIMENTAL COMPARISON OF MONOCULAR AND STEREO VISUAL FASTSLAM IMPLEMENTATIONS

Relatore: Stefano Debei

Correlatore: Marco Pertile

Riccardo Giubilato mat. 1080464

Anno Accademico 2015/2016

Contents

Chapter 1

Introduction

In planetary exploration space missions some of the most important tasks are the estimation of the robot path and the mapping of an unknown environment. The former task can be fulfilled with the help of odometric evaluations measuring the rotation of the wheels or by integrating the velocity history of the robot provided by IMU measurements. Unfortunately, dead-reckoning methods are prone to errors that grow without bound over time [13]. One of the main sources of error is the wheel slippage: the path predicted by wheel odometry can drift substantially from the actual path of the vehicle. More robust estimations of the motion are provided by visual odometry algorithms where the displacement between matching image features over subsequent frames are used to compute both the rotation and the traslation of the inboard stereo-camera. Visual odometry is still prone to errors [12] even if in a much less extent than wheel odometry errors. SLAM (Simultaneous Localisation and Mapping) algorithms can improve the motion estimates from odometry by evaluating the correlation between the vehicle motion hypothesis and the map of the environment measured by the on-board instruments as RADARs, LiDARs, sonars or cameras. SLAM can therefore fullfil both

the estimation of the robot path and the reconstruction of the landscape in which the robot is moving.

While SLAM algorithms can be implemented by using a wide variety of sensors, visual SLAM is peculiar for great robustness in data association and for the ability to acquire and update a dense map of landmarks using relatively cheap hardware. Visual SLAM algorithms can be also implemented in a wide variety of applications, from vehicle navigation to satellite docking but also in medical application.

In this work both stereo and monocular SLAM formulations are implemented to highlight the difference between the two paradigms in terms of performance and flexibility. Chapter 2 will describe the characteristics of the Simultaneous Localization and Mapping problem addressing the most famous approaches to solve it. On chapter 3 a brief review of the state of the art in SLAM implementation is made and in chapter 4 the algorithms developed in this thesis work for both monocular and stereo FastSLAM are explained. Finally, the performances of the developed algorithms are presented in chapter 5.

Chapter 2

Overview on SLAM

2.1 Simultaneous Localisation and Mapping

The Simultaneous Localisation and Mapping (SLAM) problem asks if it is possible for a mobile robot to be placed at an unknown location in an unknown environment and for the robot to incrementally build a consistent map of this environment while simultaneously determining its location within this map. The genesis of the probabilistic SLAM problem occured at the end of the 1980s thanks to the work of H. Durrant-Whyte, P. Cheeseman, R. Smith, M. Self and others researchers who highlighted the correlations between the estimations of landmarks location and the robot poses. Their work showed that when a robot moves through an unknown environment taking relative observations of landmarks, those estimations are necessatily correlated because of the common error in the knowledge of the robot pose. The idea of a "net shaped" structure of correlations between the robot pose and the map but also between the components of the map itself led to the formulation of the EKF-SLAM solution where both the robot pose and the landmarks coordinates are evaluated in a joint state formulation which is updated follow-

ing every landmark observation through an Extended Kalman Filter update. The strength of the probabilistic formulation of the SLAM problem can be found in the following consideration: the work in [4] showed that (in a linear Gaussian formulation) the correlation between landmark estimates increase monotonically as more and more observations are made leading to the convergence of the map regardless of the motion of the robot. This result can be explained considering that simultaneous observations of different landmarks provide a high correlated measurement of the relative locations. As the robot moves through the environment more measurements of relative locations of landmarks are made, updating not only the position of the actual measured landmarks but also the position of the highly correlated landmarks to the ones that have been recently observed, therefore updating a bigger portion of the estimated map.

2.1.1 Preliminaries

Before defining the parameters involved in the SLAM problem is necessary to recall some definitions of the following terms:

- *probability distribution*: function that assigns a probability to each measurable subset of the possible outcome of a random procedure. The term is often used as a synonym of the more correct *probability density function*.

- *posterior probability distribution*, $P(A|B)$: probability distribution of a random variable conditioned on the evidence obtained from previous evidences.

- *prior probability distribution*, $P(A)$: probability distribution of a random variable before any evidence is taken into account.

- *joint probability distribution*, $P(A, B)$: probability distribution that assigns the probability of each (A, B) falling into a particular range or subset
- *Bayes Theorem*: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

Let the *pose* be defined as the euclidean position in which the robot is located at time step t, $\mathbf{s}_t$. The path of the robot is therefore defined as a set of consequent pose starting from the initial position $\mathbf{s}_0$ which is arbitrarily chosen.

$$\mathbf{s}_{0:t} = \{\mathbf{s}_0, ..., \mathbf{s}_T\}$$

Let also $\mathbf{u}_t$ denote the input given to the robot to move from the current pose to the next. This information can be provided by the actual commands given to the motors but can also be obtained by the odometry of the wheels or the integration of the robot velocity using the outputs of the IMU unit if available. The history of the inputs is therefore:

$$\mathbf{u}_{0:t} = \{\mathbf{u}_0, ..., \mathbf{u}_T\}$$

Let now be defined the true location of the i^{th} landmark, $\mathbf{m}_i$, and the observation of that landmark obtained at time step t, $\mathbf{z}_{i,k}$. The set of all the landmarks is addressed as the "map":

$$\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_i\}$$

Let also be defined the set of all landmark observations:

$$\mathbf{Z_{0:t}} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_i\}$$

Solving the SLAM problem requires computing for every time step the following joint probability distribution:

$$P(\mathbf{s}_t, \mathbf{m}|\mathbf{Z_{0:t}}, \mathbf{U_{0:t}}, \mathbf{s}_0) \tag{2.1}$$

This means computing the posterior joint probability distribution of the pose of the vehicle and the location of the observed landmark as a function of knowledge of landmark measurements and inputs given to the robot at time step t. The formulation is recursive, starting from an estimate of

$$P(\mathbf{s}_{t-1}, \mathbf{m} | \mathbf{Z_{0:t-1}}, \mathbf{U_{0:t-1}}, \mathbf{s}_0)$$

the distibution at time step t is computed using Bayes Theorem taking into account the inputs $\mathbf{u}_t$ and the observations $\mathbf{z}_t$. Let now be defined the **motion model** as the probability distribution of the pose $\mathbf{s}_t$ given the previous location and the inputs given $\mathbf{u}_t$

$$P(\mathbf{s}_t | \mathbf{u}_t, \mathbf{s}_{t-1})$$

The **observation model** is the posterior probability distribution of the landmark measurement $\mathbf{z}_{i,t}$ when the actual landmark location $\mathbf{m}_i$ and the pose of the robot $\mathbf{s}_t$ are known.

$$P(\mathbf{z}_{i,t} | \mathbf{m}_i, \mathbf{s}_t)$$

A two step prediction-update procedure now takes place to compute the full posterior probability at time step t by using both the motion and the observation models.

- *Time update*: being the pose only dependent on the inputs and the last known pose of the robot, not on the map or the measurements, the state transition described by the motion model is assumed to be a Markov process. The joint *prior* distribution of $(\mathbf{s}_t, \mathbf{m})$ can therefore be written as:

$$P(\mathbf{s}_t, \mathbf{m} | \mathbf{Z_{0:t-1}}, \mathbf{U_{0:t}}, \mathbf{s}_0) = \int P(\mathbf{s}_t | \mathbf{u}_t, \mathbf{s}_{t-1}) * P(\mathbf{s}_{t-1}, \mathbf{m} | \mathbf{Z_{0:t-1}}, \mathbf{U_{0:t-1}}, \mathbf{s}_0) d\mathbf{s}_{t-1} \tag{2.2}$$

Note that the first term of the equation represents the probability distribution of $(\mathbf{s}_t, \mathbf{m})$ before any measurement at time step t is taken into account. The knowledge of the path is then computed only as a function of the motion of the robot, which is known with an uncertainty related to the odometry or to the inputs given.

- *Measurement update*: the observations obtained at time step t are now taken into account to correct the full *posterior* computed in the prediction step via Bayes Theorem:

$$P(\mathbf{s}_t, \mathbf{m}|\mathbf{Z_{0:t}}, \mathbf{U_{0:t}}, \mathbf{s}_0) = \frac{P(\mathbf{z}_{i,t}|\mathbf{m}_i, \mathbf{s}_t)P(\mathbf{s}_t, \mathbf{m}|\mathbf{Z_{0:t-1}}, \mathbf{U_{0:t}}, \mathbf{s}_0)}{P(\mathbf{z}_k|\mathbf{Z_{0:t-1}}, \mathbf{U_{0:t}})} \quad (2.3)$$

Finding a solution to the probabilistic SLAM problem requires formulating an appropriate representation for the motion and observation model that allows an efficient and consistent computation of the prior and posterior distributions in eq. 2.2 and 2.3. In the most recent years quite a few solutions of the problem have been formulated: EKF-SLAM, FastSLAM and Graph-SLAM are the most famous in today's literature. In the following sections every one of the three will be introduced.

2.1.2 EKF-SLAM

The algorithm based on extended Kalman filters is acknowledged as the most influential SLAM formulation and historically the earliest. The foundation of the algorithm is the state-space formulation of all the variables involved: the robot pose and the location of the landmarks in the map. An additive Gaussian noise is added to the state vector in order to model the uncertainty related to every component of the state. In the EKF-SLAM algorithm each probability distribution is modeled as a Gaussian distribution and the motion

and observation model are linearized. The Gaussian assumption is proven to be effective only if small uncertainties are taken into account otherwise the Kalman filter update would introduce errors leading eventually to the divergence of the filter. The spring newtork analogy is a great representation of the mechanism involved in the solution of the SLAM problem via EKF. Let a network of springs describe the correlation between landmarks in the map. The stiffness of the springs is proportional to the correlation: the higher is the correlation the higher is the stiffness of the connecting spring. When the robot moves through the environment the correlation or stiffness between simultaneously observed landmarks increases, propagating the correction in the previously known map linked by springs to the observed region. The basis of EKF-SLAM is to describe the vehicle motion in the form

$$P(\mathbf{s}_t|\mathbf{u}_t, \mathbf{s}_{t-1}) \rightarrow \mathbf{s}_t = \mathbf{f}(\mathbf{s}_{t-1}, \mathbf{u}_t) + \mathbf{w}_t$$

where $\mathbf{f}(.)$ models the vehicle kinematics and $\mathbf{w}_t$ models a zero mean white additive Gaussian noise with covariance $\mathbf{Q}_t$. The observation model is described as

$$P(\mathbf{z}_t|\mathbf{s}_t, \mathbf{m}) \rightarrow \mathbf{z}_t = \mathbf{h}(\mathbf{s}_t, \mathbf{m}) + \mathbf{r}_t$$

where $\mathbf{h}(.)$ models the observation model and $\mathbf{r}_t$ models a zero mean white additive Gaussian noise with covariance $\mathbf{R}_t$. The state vector, also addressed as *combined state vector*, comprises both the robot pose and the map and is defined as

$$\mathbf{y}_t = \{\mathbf{s}, \mathbf{m}_1, \mathbf{m}_2, ...\mathbf{m}_n\}$$

If the robot moves on a plane and every landmark state has dimension 3 then the combined state vector has dimension $3N + 3$ where N is the number of

landmarks.

$$\mathbf{y}_t = \{\mathbf{s}_x, \mathbf{s}_y, \phi, \mathbf{m}_{1,x}, \mathbf{m}_{1,y}, \mathbf{m}_{1,z}, \mathbf{m}_{2,x}, \mathbf{m}_{2,y}, \mathbf{m}_{2,z}, ...\mathbf{m}_{n,z}\}$$

EKF-SLAM algorithm computes the posterior probability distribution

$$P(\mathbf{y}_t|\mathbf{Z}_{0:t}, \mathbf{U}_{0:t})$$

where the state vector is represented by a Gaussian distribution of mean $\mathbf{y}_t$ and covariance $\mathbf{P_t}$.

$$\mathbf{P_t} = \begin{bmatrix} \mathbf{P_{ss}} & \mathbf{P_{sm}} \\ \mathbf{P_{ms}} & \mathbf{P_{mm}} \end{bmatrix}$$

The recursive computation starts with an assumption for the initial position of the robot $\mathbf{s}_0$ and the related covariance. Next the time update step takes place, updating the path on the robot without any information about the variation of the environment observed. The *a priori* knowledge of the new state is (the symbol (-) will from now on address the knowledge of a variable before any measurement correction)

$$\begin{aligned} \mathbf{s}_{t,(-)} &= \mathbf{f}(\mathbf{s}_{t-1}, \mathbf{u}_t) \\ \mathbf{P}_{ss,t,(-)} &= \nabla\mathbf{f}(\mathbf{s}_{t-1}, \mathbf{u}_t)\mathbf{P}_{ss,t-1}\nabla\mathbf{f}(\mathbf{s}_{t-1}, \mathbf{u}_t) \end{aligned} \tag{2.4}$$

where $\nabla\mathbf{f}$ is the Jacobian of the function that models the motion of the robot evaluated in $(\mathbf{s}_{t-1}, \mathbf{u}_t)$. The measurement update is used to correct the *a priori* estimate for the robot pose taking into consideration the measurement of the environment: the observed state of the landmark relative to the robot, which is dependent on the new robot location, will now reduce the uncertainty of the pose, rejecting the motion hypotesis that are highly unlikely. The time update is computed only for the robot pose if the map is assumed to be static,

SLAM in open environment with moving features is still an open research topic and will not be addressed in this thesis work. The measurement update is computed as follows

$$\mathbf{y}_t = \begin{bmatrix} \mathbf{s}_{t,(+)} \\ \mathbf{m}_t \end{bmatrix} = \begin{bmatrix} \mathbf{s}_{t,(-)} \\ \mathbf{m}_{t-1} \end{bmatrix} + \mathbf{K}_t(\mathbf{z}_t - \mathbf{h}(\mathbf{s}_{t,(-)}, \mathbf{m}_{t-1})) \tag{2.5}$$

$$\begin{aligned} \mathbf{P}_{t,(+)} &= \mathbf{P}_{t,(-)} - \mathbf{H}_t^T \mathbf{S}_t \mathbf{H}_t \\ \mathbf{S}_t &= \nabla \mathbf{h} \mathbf{P}_{t,(-)} \nabla \mathbf{h}^T + \mathbf{R}_t \\ \mathbf{W}_t &= \mathbf{P}_{t,(-)} \nabla \mathbf{h}^T \mathbf{S}_t^{-1} \end{aligned} \tag{2.6}$$

where $\mathbf{W}_t$ is the Kalman gain, $\mathbf{S}_t$ is the innovation covariance and $\mathbf{R}_t$ is the measurement covariance. Please note that the procedure explained in eq. 2.4-2.6 is a simple Extended kalman Filter update. The EKF-SLAM algorithm suffers from some severe issues that compromise its reliability in a lot of applications. Here follows a summary of the main weaknesses:

- *Non-linearity*: The EKF algorithms employs linearised models for the motion and measurement models so the convergence is guaranteed only in a linear case. High non linearity can rapidly lead to divergence of the algorithm if any safety measure is adopted.

- *Data Association*: The association of the observed landmarks to the known map is a very delicate procedure. Every time a landmark is measured, the algorithm must relate it to the previously known map to remember its previously known location. Incorrect associations lead rapidly to filter divergence especially in the case of EKF-SLAM where the correlation between all the landmarks are taken into consideration. In the paper [5] EKF-SLAM and FastSLAM are compared showing that EKF suffers from high sensibility on wrong associations.

- *Computational Effort*: The state space is subject to dimensional increase as more new landmark are observed. As explained before, the dimension of the state vector is $3N+3$ where N is the number of landmarks. The covariance P associated to the mean y is a $(3N+3)*(3N+3)$ matrix so the computational cost grows quadratically with the number of landmarks. EKF-SLAM is not suited for very dense map computations.

2.1.3 FastSLAM

The FastSLAM approach [15] for solving the SLAM problem integrates particle filters and extended Kalman filters. The foundations of FastSLAM are to be found in the work of Murphy [18]: features estimations are not correlated if the path of the robot is known with no uncertainty. This means that the landmark estimation can be approached with a single EKF update per-landmark which would require little computational cost as it require to invert a $3x3$ covariance matrix. The computational cost will then grow in time logarithmic to the number of landmarks in the map. The trajectory of the robot is of course known with some uncertainty but particle filters estimate a probability distribution sampling N values of the random variable (in this case $\mathbf{s}_t$) and assing to every one of them a weight proportional to the likelihood of that particular value. On a per-particle basis the path is therefore perfectly known. Murphy's observation allows to factor the estimation of the full probability distribution in eq. 2.1 into two separate problems, robot path and map computation:

$$P(\mathbf{s}_t, \mathbf{m}|\mathbf{Z_{0:t}}, \mathbf{U_{0:t}}, \mathbf{s}_0) = P(\mathbf{s}_t|\mathbf{Z_{0:t}}, \mathbf{U_{0:t}}, \mathbf{s}_0) \prod_{n=1}^{N} P(\mathbf{m}|\mathbf{Z_{0:t}}, \mathbf{U_{0:t}}, \mathbf{s}_0) \quad (2.7)$$

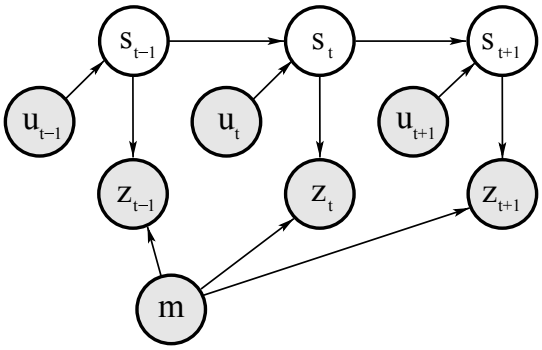


Figure 2.1: Bayesian network graphical depiction of the SLAM problem: the robot moves from position $\mathbf{s}_{t-1}$ to $\mathbf{s}_{t+1}$ as a result of the inputs $\mathbf{u}$ observing through the measurements $\mathbf{z}_{t-1}$, $\mathbf{z}_t$, $\mathbf{z}_{t+1}$ the state of the landmark $\mathbf{m}$

The posterior is decomposed into N+1 recursive estimators, one over the robot path and N over the landmark locations conditioned on the path estimate. Figure 2.1 depicts graphically the data acquisition process in form of a Bayesian network.

The prediction or time-update step is computed by a particle filter which estimates the posterior over robot path $P(\mathbf{s}_t|\mathbf{Z_{0:t}}, \mathbf{U_{0:t}}, \mathbf{s}_0)$ without assuming any type of probability distribution, the Gaussianity assumption of EKF-SLAM is infact an approximation and it represents by no means the actual distribution of the robot pose at time step t. The non-linearity of the motion model does not implicate any issue in a particle filter scenario. A set of particles is sampled by a proposed distribution which is just assumed before any correction takes place. Each particle is in the form:

$$\mathbf{S}_t^{[m]} = \left\langle \mathbf{s}_t^{[m]}, \mu_{1,t}^{[m]}, \mathbf{\Sigma}_{1,t}^{[m]}, ..., \mu_{N,t}^{[m]}, \mathbf{\Sigma}_{N,t}^{[m]}, \mathbf{w}_t^{[m]} \right\rangle \tag{2.8}$$

where $\mu_{j,t}^{[m]}$ and $\mathbf{\Sigma}_{j,t}^{[m]}$ are respectively the mean and covariance associated to the j-th landmark in the map known at time step t. Every particle contains one trajectory hypotesis and one map hypotesis. $\mathbf{w}_t^{[m]}$ is the weight associated

	Robot pose	Landmark 1	Landmark 2	...	Landmark N
Particle 1	$\{x, y, \phi\}$	$\{\mu_1, \Sigma_1\}$	$\{\mu_2, \Sigma_2\}$	...	$\{\mu_N, \Sigma_N\}$
Particle 2	$\{x, y, \phi\}$	$\{\mu_1, \Sigma_1\}$	$\{\mu_2, \Sigma_2\}$	...	$\{\mu_N, \Sigma_N\}$
...	...	...	...	...	...
Particle M	$\{x, y, \phi\}$	$\{\mu_1, \Sigma_1\}$	$\{\mu_2, \Sigma_2\}$	...	$\{\mu_N, \Sigma_N\}$

Table 2.1: Particles in FastSLAM

to the m-th particle, the role of the weight will be deeply explained later. The filter update is performed in the following steps:

1.Prediction step: sampling from the path posterior. Given the input $\mathbf{u}_t$, the pose for the m-th particle is computed drawing a sample from the motion posterior

$$\mathbf{s}_t^{[m]} \sim P(\mathbf{s}|\mathbf{s}_{t-1}^{[m]}, \mathbf{u}_t)$$

where $\mathbf{s}_{t-1}^{[m]}$ is the pose at time step t-1 according to the path hypotesis of the m-th particle. The resulting sample is stored in a temporary set of particles along with the path of previous poses. The motion model by no means have to be formulated as a Gaussian distribution, in figure 2.2 are depicted few options on how to draw samples according to the motion of the robot and the uncertainty associated with it but also on eventual a priori knowledge of the scenario in which the robot is moving. The motion model can also take into account not the input given to the robot but also the odometry data coming from the wheels or from a GPS track if available or from the velocity information given by an IMU unit. In the resampling step the most likely trajectories are resampled to build a final set of particles, if no resampling is performed the M path hypotesis diverge monotonically as depicted in figure 2.3.

2.Updating the landmark estimate Next, FastSLAM algorithm updates the posterior over the landmark estimates that are observed at time step t, represented by the mean $\mathbf{x}_{t,j}^{[m]}$ and the covariance $\mathbf{\Sigma}_{t,j}^{[m]}$. The updated values are then added to the temporary particle set, along with the new pose. The non observed landmarks are not evaluated in this step. For the observed features an extended Kalman filter updated is performed. Being $h(\mathbf{s}_t, \mathbf{m}_j)$ the measurement model and $H(\mathbf{s}_t, \mathbf{m}_j)$ its Jacobian evaluated for the current pose and the j-th landmark

$$\begin{aligned} \mathbf{x}_{j,(+)}^{[m]} &= \mathbf{x}_{j,(-)}^{[m]} + \mathbf{K}_{t,j}^{[m]}(\mathbf{z}_j - \mathbf{h}(\mathbf{s}_t, \mathbf{x}_{j,(-)}^{[m]}) \\ \mathbf{\Sigma}_{(+),j}^{[m]} &= (\mathbf{I} - \mathbf{K}_{t,j}^{[m]}\mathbf{H}_{t,j}^{[m]T})\mathbf{\Sigma}_{(-),j}^{[m]} \\ \mathbf{K}_{t,j}^{[m]} &= \mathbf{\Sigma}_{(-),j}^{[m]}\mathbf{H}_{t,j}^{[m]}\mathbf{S}_{t,j}^{[m]-1} \\ \mathbf{S}_{t,j}^{[m]} &= \mathbf{H}_{t,j}^{[m]T}\mathbf{\Sigma}_{(-),j}^{[m]}\mathbf{H}_{t,j}^{[m]} + \mathbf{R}_t \end{aligned} \tag{2.9}$$

Step 1 and 2 are repeated M times resulting in a set of M temporary particles.

3. Resampling In a final step, FastSLAM resamples this set of particles drawing from this temporary set M particles with replacement, therefore a single particle can be resampled multiple times until the set of M new par-

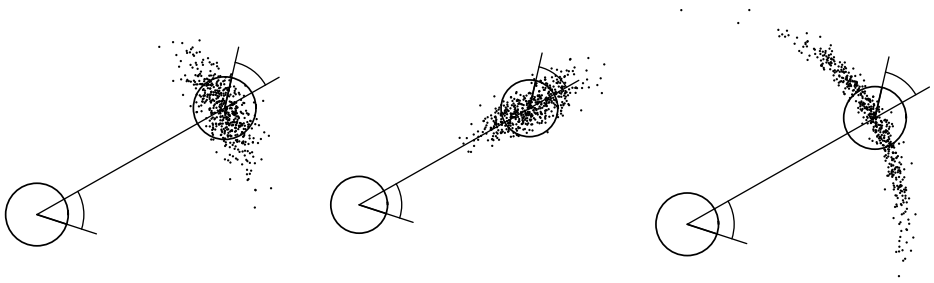


Figure 2.2: Depiction of the sampling process, according to the motion model of the robot a set of N path samples is drawn.

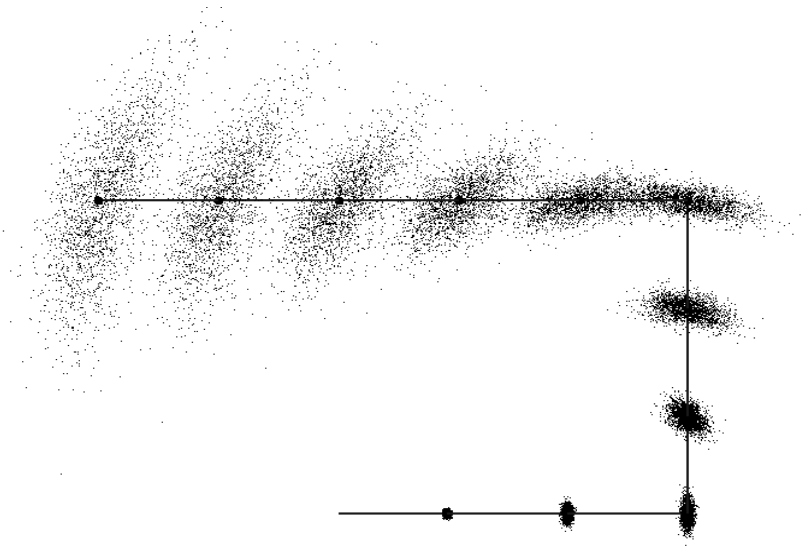


Figure 2.3: Sparsification of the trajectories if no correction based on observation is computed

ticles is complete. The new particle set will be distributed according to the proper probability distribution of the poses which can differ substantially from the proposal distribution assumed to sample the temporary set of particles. The situation is illustrated in figure 2.4 where the solid line is the actual distribution and the dashed line is the proposal distribution, the samples drawed from the proposal are weighted during the landmark estimation update to better approximate the actual distribution of the poses taking into account the current observations of the robot. The higher the weight the more likely is a particle to resemble the actual target distribution. The temporary set of particles is then resampled (with replacement) according to the weights or *importance factors* of the particles. Weigths are computed for every landmark update as follows:

$$\mathbf{w}_j^{[m]} = \frac{1}{\sqrt{2\pi|\mathbf{S}_{t,j}^{[m]}|}} \exp\{-\frac{1}{2}(\mathbf{z}_j - \hat{\mathbf{z}}_j^{[m]})^T \mathbf{S}_{t,j}^{[m]-1}(\mathbf{z}_j - \hat{\mathbf{z}}_j^{[m]})\}$$

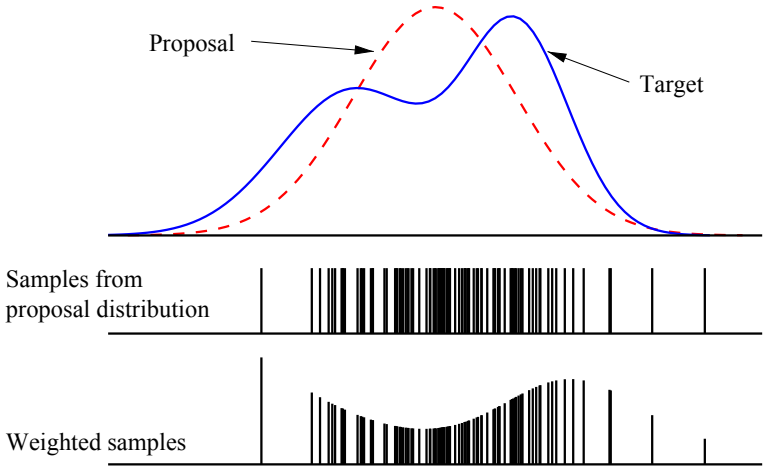


Figure 2.4: Target and proposal distribution of the robot poses. Samples from the proposal distribution are weighted according to eq 2.10

where $\mathbf{S}_{t,j}^{[m]}$ is the *innovation covariance* computed during the landmark estimation update and $\hat{\mathbf{z}}_j^{[m]}$ is the result of the measurement model, that is the measurement that the observer expect to take given a certain position and a ceraint landmark.

2.1.4 Graph-SLAM

Graph-SLAM is another well known paradigm of the SLAM problem by S. Thrun and M. Montemerlo [16]. The main idea behind GraphSLAM is that robot motions and measurements can be represented by a graph where the nodes are vehicle and landmarks positions and the edges linking each node are costraints based on the negative log likelihood of each connected node. Those likelihoods are directily computed using the motion and the measurement models, the former is used to compute the costraint between consequent robot poses while the latter is used to compute the costraint between a landmark position and the robot pose from which the landmark was observed. The sum

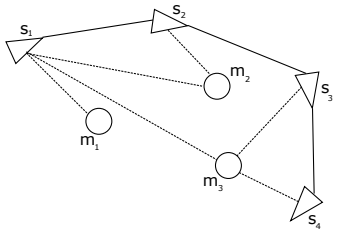


Figure 2.5: Graph representation of the robot pose and the map. The links from pose to pose or pose to landmark are associated to a costraint in the position (i, j) of the information matrix where the i-th row is related to the first node and the j-th row is associated to the second node

of all the costraints leads to a *least squares minimization* problem, naively the target function of GraphSLAM is to minimize this sum to get the most likely map and the most likely robot path. Figure 2.5 depicts the graph representation of the problem. The costraint between two robot poses is

$$(\mathbf{x}_t - g(\mathbf{x}_{t-1}, \mathbf{u}_t))^T \mathbf{Q}_t^{-1} (\mathbf{x}_t - g(\mathbf{x}_{t-1}, \mathbf{u}_t))$$

where g is the motion model. The costraint between a robot pose and a landmark location is

$$(\mathbf{z}_{t,i} - h(\mathbf{m}_i, \mathbf{s}_t))^T \mathbf{R}_t^{-1} (\mathbf{z}_{t,i} - h(\mathbf{m}_i, \mathbf{s}_t))$$

The sum of all the costraints is

$$\begin{aligned} J_{sum} &= \mathbf{x}_0^T \mathbf{\Omega}_0 \mathbf{x}_0 + \\ &= \sum_t (\mathbf{x}_t - g(\mathbf{x}_{t-1}, \mathbf{u}_t))^T \mathbf{Q}_t^{-1} (\mathbf{x}_t - g(\mathbf{x}_{t-1}, \mathbf{u}_t)) + \\ &= \sum_t \sum_i (\mathbf{z}_{t,i} - h(\mathbf{m}_i, \mathbf{s}_t))^T \mathbf{R}_t^{-1} (\mathbf{z}_{t,i} - h(\mathbf{m}_i, \mathbf{s}_t)) \end{aligned}$$

where Ω is the information matrix which contains the values of the costraints. Let also be ξ the information vector which contains the values associated to

the nodes. In the associated information matrix the off-diagonal elements are all-zero with few exceptions: a non-zero value will be located in (i, j) where the i-th row is associated to a robot pose $\mathbf{s}_{t-1}$ and the j-row is associated to the next robot pose $\mathbf{s}_t$, the link between those nodes represents the control $\mathbf{u}_{t-1}$. The other non-zero elements are the links between a map feature $\mathbf{m}_j$ and the robot pose $\mathbf{s}_t$ from where the landmark is observed. After the information matrix is assembled the path and the map can be obtained from the linearization of $\mathbf{\Omega}$ and ξ via the following equations where $\mathbf{x}$ is the map estimate:

$$\begin{aligned} \mathbf{x} &= \mathbf{\Sigma}\xi \\ \mathbf{\Sigma} &= \mathbf{\Omega}^{-1} \end{aligned}$$

If the landmark are observed only on a single time step each, the information matrix can be reordered so it becomes a diagonal matrix, thus the computational time required for the inversion is linear to the length of the information vector. Normally every feature is observed over multiple time steps with large time delays between every observation. The matrix is then much more complex and the reordering can be difficult. The GraphSLAM algorithm employs a *variable elimination algorithm* to compute the information matrix inversion to extract information just about a certain number of elements in the information vector as the path of the robot instead of the landmark state. Thinking about the graph as a mass and springs network, the algorithm performs an elimination of some links establishing new ones only between the requested nodes, mantaining invariant the total force applied to the masses. A much smaller matrix is now requested to be inverted thus a significant computational time reduction is obtained. An analogue

procedure is then performed to extract map features from $\mathbf{\Omega}$ and ξ.

Chapter 3

Related Work

In this chapter a brief summary is provided about the most important and influential works on the topic of vision based SLAM or SLAM implemented using camera devices as sensors.

3.1 Monocular SLAM

One of the first implementations of real time monocular visual SLAM is **"MonoSLAM"** by **Davison** (2007) [23]. The algorithm computes a sparse map of very consistent features using the EKF paradigm to obtain real-time 30 Hz localization over a small area. The hardware used is a very inexpensive IEEE 1394 webcam with a wide angle lens. From a single image no information about depth can be obtained. The problem of localizing the observed features tridimensionally is one of the main challenges in monocular vision and many approaches have been employed in the following works. In MonoSLAM every features is initialized as a 3D line starting from the camera optical center and pointing towards the projection of the feature on the image plane. A set of discrete depth hypotesys is distributed along this line and the

likelihood of them to better resemble the actual depth is computed over the following time steps on the approximation that the next observations only carry information about the depth, not about the direction. When the probability distribution of the depth is sufficiently peaked the feature is converted to *fully initialized* with a standard 3D Gaussian distribution. Feature detection is performed with the Shi and Tomasi detector and a very robust feature matching algorithm is employed. To extend the viewing angle and scale from which a feature can be recognized, for every featured detected a small image window centered on it is stored. Given the estimated trajectory between following time steps those little image portions are warped according to the change in parallax and scale and then projected on the sensor plane. This work showed that real time monocular SLAM is feasible even on inexpensive hardware but the map obtained is very sparse and the drift-free tracking is permitted by EKF-SLAM which can easily diverge from false data association, more likely during dense map evaluations, and suffers from quadratical computational time increase in the number of landmarks acquired.

Kwok, **Ha** and **Fang** (2007) [6] developed a cost function based method for data association in bearing-only visual SLAM. While in range and bearing data acquisition the association of the observed landmarks can be performed by Mahalanobis distance between the known and measured state to select the most likely association, in bearing only SLAM within a measure no depth information is obtained. During conventional data association, the Mahalanobis distance is computed on the innovation error, that is the difference between the known state and the expected state of the landmark during a measure, and the innovation covariance, which include the uncertainties on the known state and on the measure. This work takes into account the uncertainty on the state of the landmark and the uncertainty on the bearing

which is the actual measure. The likelihood of an association is computed for every landmark as a function of their predicted bearing in relation to the camera displacement. A cost function is computed and an association is declared between the candidates that give a minimal cost.

Kootstra, **de Jong** and **D. Wedema** (2009) [5] worked on a comparison of EKF-SLAM and FastSLAM approaches to highlight the issues related to the standard extended Kalman filter paradigm. A Pioneer 2 DX robot with a single camera was operated to repeatedly observe the same environment moving on a loop. Two settings for the matching algorithm are considered, an unreliable and a reliable one. The difference between those two sets is the probability of establishing false data association purely on the visual properties of the detections. A buffer of a fixed number of images is stored, only when a feature is been observed on a minimum number of frames it is considered an actual measure. Depth information are obtained through triangulation over the location of the feature in the image buffer. The results of this work showed that FastSLAM outperformed EKF-SLAM both on reliable and unreliable settings. EKF-SLAM was proven to suffer much more than FastSLAM on wrong associations because of the single hypotesis tracking. Every correction on landmark states propagates on the robot pose over great distances therefore a wrong association induces an error over the entire pose history of the robot. In the FastSLAM scenario the effect of a false association is much more retained because the pose estimation relies on sampling from a proposed particle cloud.

Solï¿½, **Monin Devy** and **Lemaire** (2008) [8] investigated the topic of undelayed initialization of landmarks in bearing-only SLAM. The main intuition behind this approach is that when a new feature is observed, bearing information can still be useful to recuce the uncertainty of the direction of

the robot while they carry no information about the translations. The approach used is to formulate the landmark hypotesis as a series of Gaussians, selecting the more likely ones in the following steps and then initializing the landmark as a 3D Gaussian distribution.

A different solution to undelayed initialization is given by **J. Civera**, **A. J. Davison** and **J. M. Martinez Mondiel** (2008) [17]. In this work the problem of extracting depth information while using the landmark measurement to correct the robot pose and orientation is solved by using a different parametrisation for the state of the landmark. In this work it is proven that a gaussian distribution doesn't accurately represent the actual probability distribution of a landmark at the time of the first acquisition because naively it should lie on a cone centered on the landmark projection the camera sensor. An inverse depth parametrization is proposed: landmark are described using a 6 element vector which contains the euclidean pose of the optical center at the time of the first acquisition, the two angles describing the landmark direction and the depth. When a landmark is observed through a certain parallax change (for example 3 degrees) the uncertainty about the landmark resembles very well a Gaussian 3D covariance and then the landmark is converted to an Euclidean parametrization. This conversion is not necessary but can improve the computational time. While parametrized in inverse depth landmarks can provide useful information about the orientation of the camera.

M. Li, **B. Hong**, **Z. Cai**, **R. Luo** (2008) [11] formulated a variant of the common particle filter SLAM algorithm (as FastSLAM) where the prediction step takes into account the current set of measurement to draw the most likely particle before the landmark state update is performed refining the initial particle sampling to better approximate the true distribution (a

similar process is implemented in FastSLAM 2.0 by Thrun and Montemerlo). Their results show not only a better pose estimate but also a more consistent map acquisition.

C. Gamallo, **M. Mucientes** and **C. Regueiro** (2009) [22] investigated the use of an omnidirectional camera (fish-eye lens) in a monocular FastSLAM environment. The approach used is based on FastSLAM 2.0 where the temporary particle sampling is performed considering the current measurements to obtain a more likely set of poses than the one obtained only sampling from the motion model. The camera mounted on the robot is equipped with a infrared filter as the landmark observer are the lights on the ceiling. The environment is a museum therefore the camera can observer a big portion of the whole map having a 185°field of view.

3.2 Stereo SLAM

One of the first work on robot localization and map building using stereo vision is from **S. Se**, **D. Lowe** and **J. Little** (2006) [10]. SIFT feature descriptor (Lowe, 1999) is used for the consistent performances in data association over high scale and view angle variations. Previous approaches were based on Harris corner detector which is very sensitive to scale and therefore is not suited for mapping algorithms. SIFT features are matched between the two cameras satisfying the criteria of epipolar costraint, disparity and orientation costraint (costraint based on the proximity of correct matches), unique match costraint (ambiguous matches are discarded). In this early work, landmark state prediction from robot odometry is employed to obtain a more efficient data association from feature descriptors, no method is implemented to reduce the map uncertainty once landmarks are repeat-

edly observed. A least-squares minimization is then employed to find the robot motion that minimize the reprojection error of matching features. No Kalman data fusion is employed to reduce the path uncertainty.

The first implementation of EKF-SLAM using a stereo camera mounted on a robot is from **Davison** and **Murray** (2002) [19]. Their works addressed the feasibility of real-time SLAM using active vision (ability to rotate the cameras relatively to the robot), automatic map maintenance and goal-directed steering. Landmarks are parametrized as 3D points in the reference frame of the stereocamera and data association is performed searching candidate landmarks in the uncertainty ellipsoid of the known map. Shi and Tomasi detector is implemented (application of the Harris corner detector) to acquire new visual features, matching is performed computing the sum of squared differences between descriptors (SSD) and an epipolar costraint is applied to discard wrong matches between the two images. Two failure modes are observed, one arises from false data association and one arises from non-linearities. When uncertainty in the map is very large, measurements can induce unpredictable EKF updates which propagates on both path and map estimations. Great attention is put into selecting consistent features since a minimum number of two is sufficient for providing a fully-costrained robot position estimate. Bad landmarks are deleted from the map if their detection is considered a failure according to a set criteria. Robot automatic motion and obstacle avoidance is then addressed.

Rao-Blackwellized particle filters are commonly implemented to avoid the unconsistency of traditional EKF-SLAM, **Elinas** and **Little** [9] developed an algorithm called σSLAM that uses a particle filter to refine the robot pose and EKF to correct landmark positions in the map. The main difference from the FastSLAM framework is related to the initial sampling of the

temporary particle set. While in FastSLAM the proposal is sampled from the motion model, in σSLAM the proposal is sampled from the output of a visual odometry algorithm. A 2D occupancy grid map is also computed from the stereo image set for path planning and obstacle avoiding.

A. Gil, **O. Reinoso**, **M. Ballesta**, **D. Ubeda** (2006) [21] developed a SLAM algorithm based on Rao-Blackwellized particle filters using a stereo camera head. SIFT feature descriptors are used. A Mahalanobis distance method for data association is proposed. Commonly, Mahalanobis distance is implemented to evaluate the association between vectors with three-dimensional covariance. This work showed that Mahalanobis distance can be employed to improve the association between feature descriptors if the elements in the 128-dimensional vector are assumed to be independent. Roughly a 10% improve is obtained.

More recent approaches on stereo SLAM implementations focuse on higher data volume and map density to obtain a more complex and complete map reconstruction. **C. Brand**, **M. J. Schuster**, **H. Hirschmuller** and **m. Suppa** (2015) [24] developed an incremental graph-SLAM method where a number of dense submaps are individually computed and fused together to obtain a full environment representation. Features are extracted not only according to their visual properties but also to geometric properties in order to obtain a more robust association even on high viewpoint and scale variations.

Other approaches involve the evaluation of other image properties as photoconsistency of high-contrast pixels, corners and edges rather than feature properties. LDS-SLAM by **J. Engel**, **J. Stuckler** and **D. Cremers** (2015) [3] estimates depth from pixels with high intensity gradient reconstructing a semi-dense depth map real-time. Concurrently, it tracks the rigid-body

motion through photometrix alignment of images based on the depth maps.

Chapter 4

FastSLAM implementation

In this chapter the structure of the stereo and mono algorithms are discussed and the procedures of feature detection and matching, landmark triangulation and data association are individually discussed with the help of pictures and pseudocodes.

4.1 Preliminaries

A stereo system comprises two cameras denoted with the numbers 1 and 2 for the left and right one respectively. Each camera has a corresponding frame of reference with its z axis aligned to the optical axis, the origin coincident to the optical center, the x and y axis parallel to the sensor plane and directed according to the pixel enumeration order in the image, see figure 4.1.

4.1.1 Pinhole model

Considering the model of each camera [1], the position of a point feature comprised in the field of view of both cameras can be written as:

$$\mathbf{X}_i = \{X, Y, Z\}^T =^i \lambda_i \{x_i, y_i, 1\}^T = \lambda_i \mathbf{x}_i \tag{4.1}$$

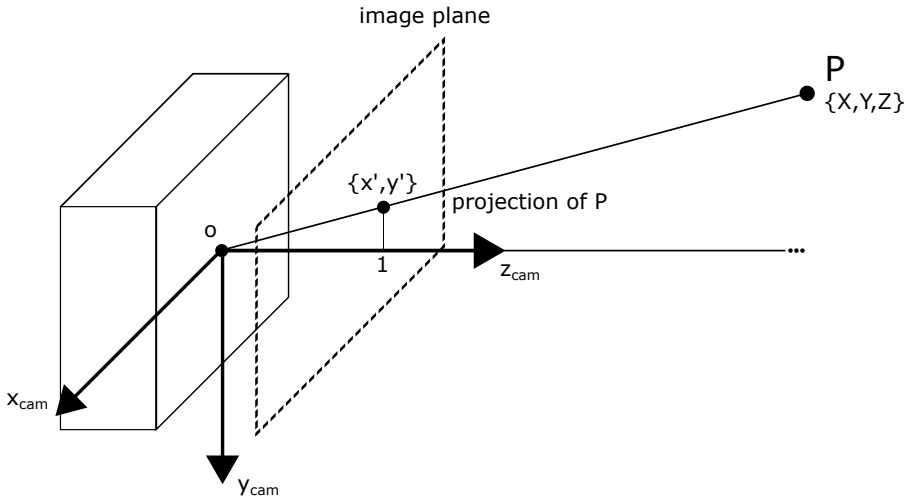


Figure 4.1: Camera reference frame

where i is 1 or 2 depending on which camera the landmark $\mathbf{X}$ is observed. $\mathbf{x}_i$ is the projection of the landmark on the image plane, which is parallel to the camera sensor and displaced from the optical center by a unitary value. λ_i is a scalar parameter associated with the depth of the feature. The relationship between the Euclidean parametrization of a landmark and its projection on the image plane is formulated according to the *pinhole model*. A thin lens is a mathematical model defined by an optical axis, a focal plane perpendicular to the axis and an optical center, the intersection between the plane and the axis. Two parameters define the thin lens: the focal length f and its diameter D. Two properties emerge: the first is that all rays entering the lens parallel to the optical axis intersect the axis at a distance f from the optical axis. The second properties is that all rays intersecting the optical center are undeflected. The fundamental equation of the thin lens is

$$\frac{1}{Z} + \frac{1}{z} = \frac{1}{f} \tag{4.2}$$

If the aperture D of a thin lens decrease to zero, the only points that contribute to the irradiace at a point on the camera sensor are on a line through

the center of the lens. The relations between the Euclidean coordinates of a feature and its projections on the image plane are:

$$\begin{aligned} x &= f\frac{X}{Z} \\ y &= f\frac{Y}{Z} \end{aligned} \tag{4.3}$$

In literature, eq 4.3 is more often found with a negative sign in front of both the fractions, that is because normally the image crossing a thin lense is projected upside down. This effect can be neutralized simply flipping the image, this corresponds to placing the image plane in front of the optical center. It is now necessary to establish a relationship between pixel and metric coordinates. The first step is to define a scaling matrix which takes into account the metric pixel dimension to transform normalized coordinates (x, y) into pixel coordinates.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.4}$$

However x' and y' are related to a reference frame that has its origin in the optical center while the pixel index count on an image (i, j) is conventionally located in the upper left corner. A translation of the origin of the reference

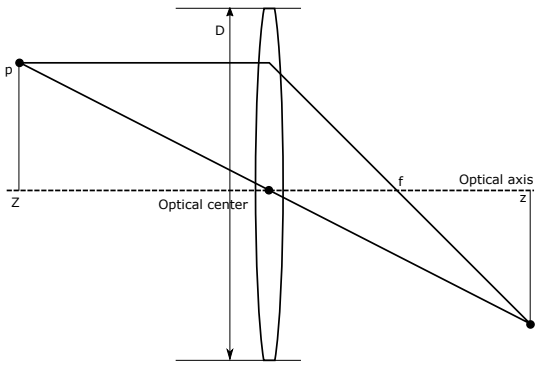


Figure 4.2: Scheme of a thin lens

frame is therefore needed. Omitting a few mathematical passages, the geometric relationship between a point of coordinates $\{X, Y, Z\}$ relative to the camera frame and its projection coordinates in pixels $\{x', y', 1\}$ is

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x f & 0 & x_0 \\ 0 & S_y f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4.5}$$

where the *intrinsec parameters* $S_x f$, $S_y f$, x_0 and y_0 are obtained experimentally via camera calibration. The matrix of the intrinsec parameters is often referred as the *calibration matrix*.

4.1.2 Triangulation

A stereo setup is also characterized bt its extrinsic parameters. To perform triangulation and determine 3D position of the scene points, the relative position and rotation between the two cameras have to be known. There parameters are named extrinsic and should be experimentally evaluated before any measurement with the stereo system can be performed. Triangulation is implemented using the algorithm of the middle point as in [13]. Given the preimage lines (the line that starts from the optical center of the camera and project the 2D image feature to the 3D real landmark coordinates) of a feature $\mathbf{x}_i$ for both camera 1 and camera 2, the algorithm finds the couple of points $\{\mathbf{X}_{s,1}, \mathbf{X}_{s,2}\}$ at the minimum distance from each line and computes the middle point of the segment.

Being ${}^1\mathbf{X}_1$ and ${}^2\mathbf{X}_2$ the vectors representing the same point in the environment referred respectively to the reference of the first and the second camera,

they can be expressed as follows:

$$\begin{aligned} {}^1\mathbf{X}_1 &= \lambda_1 \mathbf{x}_1 \\ {}^2\mathbf{X}_2 &= \lambda_2 \mathbf{x}_2 \end{aligned} \tag{4.6}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are the projection of the point X on the image plane of both cameras. It is now needed to refer both vectors to the same reference, the first camera reference frame is choosen freely.

$$\begin{aligned} {}^1\mathbf{X}_1 &= \lambda_1 \mathbf{x}_1 \\ {}^1\mathbf{X}_2 &= \lambda_2 \mathbf{R}_2^1 \mathbf{x}_2 + {}^1\mathbf{P}_{21} \end{aligned} \tag{4.7}$$

where $\mathbf{R}_2^1$ is the rotation matrix from frame 2 to frame 1, and ${}^1\mathbf{P}_{21}$ is the origin of frame 2 with reference to the origin of frame 1 and expressed in frame 1. Points $\mathbf{X}_{1,s}$ and $\mathbf{X}_{2,s}$ are calculated minimizing the following cost function:

$$g = \| {}^1\mathbf{X}_1 - {}^1\mathbf{X}_2 \|^2$$

The following values of $\lambda_{1,s}$ and $\lambda_{2,s}$ are obtained:

$$\begin{aligned} \lambda_{1,s} &= -\frac{(x_1^T R_2^1 x_2)(x_2^T\ {}^2P_{12}) + (x_2^T x_2)(x_1^T\ {}^1P_{21})}{(x_1^T x_1)(x_2^T x_2) - (x_1^T R_2^1 x_2)^2} \\ \lambda_{2,s} &= -\frac{(x_1^T x_1)(x_2^T\ {}^2P_{12}) + (x_1^T R_2^1 x_2)(x_1^T\ {}^1P_{21})}{(x_1^T x_1)(x_2^T x_2) - (x_1^T R_2^1 x_2)^2} \end{aligned} \tag{4.8}$$

where ${}^1P_{12} = -{}^1P_{21}$. ${}^2P_{12}$ is the origin of frame 1 with reference to the origin of frame 2 and expresse in frame 2. Next the points of minimum distance between the two preimage lines are computed as follows:

$$\begin{aligned} {}^1\mathbf{X}_{1,s} &= \lambda_{1,s} \mathbf{x}_1 \\ {}^1\mathbf{X}_{2,s} &= \lambda_{2,s} \mathbf{R}_2^1 \mathbf{x}_1 + {}^1\mathbf{P}_{21} \\ {}^1\mathbf{X}_1 &= \frac{{}^1\mathbf{X}_{1,s} + {}^1\mathbf{X}_{2,s}}{2} \end{aligned} \tag{4.9}$$

4.1.3 Uncertainty estimation of triangulated landmarks

After the triangulation is computed it is necessary to estimate the uncertainty on the 3D points. The following method is based on the assumption that the distribution of ${}^1\mathbf{X}$ is Gaussian. Let be f the function that gives the triangulated point in Euclidean coordinates:

$$ {}^1\mathbf{X} = f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{K}_1, \mathbf{K}_2, \mathbf{R}_2^1, {}^1\mathbf{P}_{21}) \tag{4.10} $$

The uncertainty on extrinsic and intrinsic parameters can be evaluated by stereo setup calibration and camera calibration respectively. On the assumption that every input parameter on eq. 4.10 is independent from each other, the input covariance matrix can be written as follows:

$$ \mathbf{P}_{in} = \begin{bmatrix} \sigma_{u1}^2 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_{u2}^2 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_{v1}^2 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \sigma_{v2}^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \sigma_f^2 \end{bmatrix} $$

The matrix is diagonal because of the uncorrelation of the intrisic and extrinsic parameters. The function f is non-linear therefore to propagate the uncertainty on ${}^1\mathbf{X}$ it is needed to compute numerically the Jacobian matrix of f using finite differences to approximate the derivative of every component of the function. The Jacobian matrix of f is a $(3x18)$ matrix being 3 the component of ${}^1\mathbf{X}$ and 18 the total input parameters.

$$ \mathbf{J} = \begin{bmatrix} \frac{\partial X}{\partial u_1} & \frac{\partial X}{\partial u_2} & \frac{\partial X}{\partial v_1} & \frac{\partial X}{\partial v_2} & \cdots & \frac{\partial X}{\partial f} \\ \frac{\partial Y}{\partial u_1} & \frac{\partial Y}{\partial u_2} & \frac{\partial Y}{\partial v_1} & \frac{\partial Y}{\partial v_2} & \cdots & \frac{\partial Y}{\partial f} \\ \frac{\partial Z}{\partial u_1} & \frac{\partial Z}{\partial u_2} & \frac{\partial Z}{\partial v_1} & \frac{\partial Z}{\partial v_2} & \cdots & \frac{\partial Z}{\partial f} \end{bmatrix} $$

$J_{i,j}$ is thus computed as follows:

$$ \mathbf{J}_{i,j} = \frac{f_i(x) - f_i(x + \sigma_j)}{\sigma_j} $$

That means, two output of $f(x)$ are computed, the first is computed on the input for which the uncertainty has to be estimated, the second is sligthly perturbated just on the value of the j-th parameter. Thus the covariance for ${}^1\mathbf{X}$ is

$$\mathbf{P}_{out} = \mathbf{J}\ \mathbf{P}_{in}\ \mathbf{J}^T \tag{4.11}$$

4.1.4 Epipolar costraint

Let be frame 1 and frame 2 a couple of reference frames associated to each camera of a stereo setup or to the same camera evaluated on distinct time steps. Frame 1 and 2 are oriented and positioned according to the Euclidean transformation $g = (R, t) \in SE(3)$. Let be $\mathbf{X}_1$ and $\mathbf{X}_2$ the 3D coordinates of a point relative to camera frame 1 and 2. Now let be $\mathbf{x}_1$ and $\mathbf{x}_2$ the homogeneous coordinates of the point projected in the two frames. From eq. 4.7 the two euclidean vectors can be related in the following equation:

$$\lambda_2 \mathbf{x}_2 = R\lambda_1 \mathbf{x}_1 + t \tag{4.12}$$

Let be the matrix $\hat{T}$ be defined as:

$$\hat{T} = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

thus $\hat{T}\mathbf{x} = t \times \mathbf{x}$. Eq 4.12 can written eliminating the depths by premultiplicating both sides by $\hat{T}$.

$$\lambda_2 \hat{T}\mathbf{x}_2 = \hat{T}R\lambda_1 \mathbf{x}_1$$

Since $\hat{T}\mathbf{x}_2$ is perpendicular to $\mathbf{x}_2$, the dot product $\mathbf{x}_2^T \hat{T}\mathbf{x}_2$ is equal to zero therefore $\mathbf{x}_2^T \hat{T}R\lambda_1 \mathbf{x}_1$ is zero. Being λ_1 a positive scalar value the following

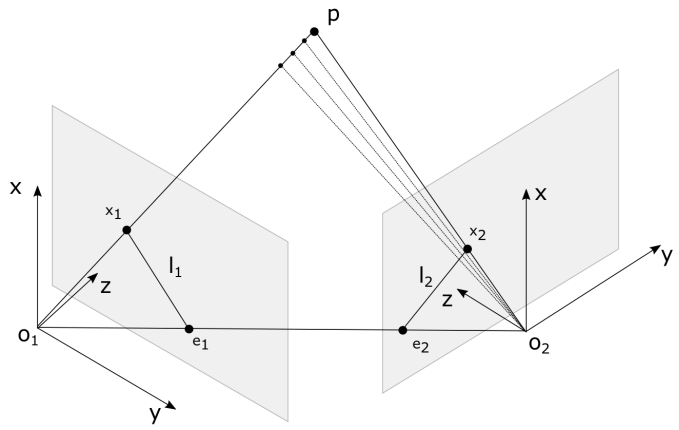


Figure 4.3: Projection of the point p in camera frames 1 and 2. The lines l_1 and l_2 are called *epipolar lines*, the lines e_1 and e_2 which are the intersections of the two image planes with the line (o_1, o_2) are called *epipoles*

result is proven:

$$\mathbf{x}_2^T \hat{T} R \mathbf{x}_1 = 0 \tag{4.13}$$

where $E = \hat{T}R$ is referred to as the *essential matrix* which encodes the relative position and orientation of the two camera poses and it is bilinear in the coordinates $\mathbf{x}_1$ and $\mathbf{x}_2$. A representation of the epipolar geometry of frames 1 and 2 is depicted in figure 4.3. Since the projections $\mathbf{x}_1$ and $\mathbf{x}_2$ are known by their pixel coordinates in the camera sensor, from eq 4.5 the

following equation can be written:

$$\begin{aligned} \mathbf{x}'_1 &= K_1\mathbf{x}_1 \\ \mathbf{x}'_2 &= K_2\mathbf{x}_2 \end{aligned} \tag{4.14}$$

where $\mathbf{x}_1 = \frac{1}{\lambda_1}\mathbf{X}_1$ and $\mathbf{x}_2 = \frac{1}{\lambda_2}\mathbf{X}_2$ and K is the calibration matrix.
Substituting $\mathbf{x} = K^{-1}\mathbf{x}'$ in eq 4.13 the following equation is obtained

$$\mathbf{x}_2'^T\ K_2^{-1,T}\ \hat{T}\ R\ K_1^{-1}\ \mathbf{x}'_1\ = 0$$

and if $F = K_2^{-1,T}\ \hat{T}\ R\ K_1^{-1}$ the equation can be written as

$$\mathbf{x}_2'^T\ F\ \mathbf{x}'_1 = 0 \tag{4.15}$$

F is called the *fundamental matrix* and is also a function of the essential matrix E by the relation $F = K_2^{-1,T}\ E\ K_1^{-1}$.
The epipolar costraint expressed in terms of pixel coordinates (eq 4.15) will be used to filter the feature matches in a stereo camera implementation discarding the ones that do not satisfy it.

4.1.5 RANSAC algorithm

RANSAC (RANdom SAmple Consensus) [2] is a paradigm for fitting a model to sets of data which contain both *inliers* and *outliers* that are values respectively represented by a unique mathematical model or not represented by that model. RANSAC is a flexible approach to find the best fitting model to a dense family of data which is the model that obtains the minimum error computed on the whole set. Algorithm 1 summarizes a general implementation of the RANSAC paradigm.

Algorithm 1 RANSAC generic implementation

1: *data* ← set of observed data ▷ Inputs
2: *model* ← model to fit on data
3: n ← number of data to be fitted by the model
4: k ← number of iterations
5: t ← threshold value to identify data represented by the computed model
6: d ← number of minimum inliers required to select a good model
7: *iteration* ← 0 ▷ Main loop
8: *best error* ← Inf
9: *best inliers* ← empty
10: **while** *iteration* $< k$ **do**
11: *inliers* ← random n values sampled from *data*
12: *model* ← model obtained by fitting *inliers*
13: **for** i = 1 : values in *data* not sampled before **do**
14: *err fit* ← error computed applying *model* to value(i)
15: **if** *err fit* $< t$ **then**
16: add value(i) to *inliers*
17: **end if**
18: **end for**
19: **if** length(*inliers*) $> d$ **then**
20: *better model* ← model parameters that satisfy the minimum consensus
21: *better error* ← error computed applying *better model* to *inliers*
22: **if** *better error* $<$ *best error* **then**
23: *best error* ← *better error*
24: *best inliers* ← *inliers*
25: **end if**
26: **end if**
27: **end while**

Data is a set of measurements or observations which should be described by a mathematical model. *model* is known except for the numerical values of its functional parameters. In *Data* are included inliers and outliers which are returned from the algorithm at the end of the loop with the best fitting model. The loop starts sampling a number of values from *data* to compute a model which fits them, the number of samples n is a function of the fitting procedure: it can be fixed (as the number of variables in the model) or can be higher than a minimum if a least squares estimation is employed. A temporary inliers set is thus built. Next every other value in *data* which has not been previously sampled is evaluated according to the estimated model, if that value is sufficiently represented by the model it is considered as another inlier and added to the temporary inliers set. If the number of temporary inliers is higher than a minimum required consensus d (which can be defined as a percentage of the total *data* length) the model becomes a candidate to be selected as a representative one and the temporary inliers set becomes the final inlier set. The error obtained applying the best model (which satisfies the minimum consensus) to all the values in *data* is stored and every time a new model returns a lower error the previous one is discarded and the new one, with the inlier set, is stored. This error can be evaluated in many ways according to the implementation of RANSAC. Algorithm 1 stops after a number of iterations which can not be sufficient for a precise estimation of the best fitting model or differently can be much higher using more computational time that needed. The generic algorithm can be improved for instance by comparing the best error to a threshold which defines the accepted error to consider "good" a model. Thus the loop can stop after less iterations or more.

4.2 Feature Detection and Matching

In this sections are addressed the procedures for image features detection, feature matching between the two image captured by the stereo camera setup and association of the current detected features to the previously known set of descriptors.

4.2.1 Feature detector

The process of feature detecting on an image is the basis for all feature-based vision sensing algorithms. A feature detector must perform a consistent and repeatable tracking of meaningful pixel regions given a set of images. One of the first solution for efficient feature detecting is from **Harris** & **Stephens** (1988) [7]. The algorithms is an improvement of the previous Moravec's corner detector and it functions evaluating the variations of image intensities, denoted by $I(x, y)$, on a fixed pixel region. The gradient of the image intensity is expanded analitically about the shift origin as follows:

$$\begin{aligned} \mathbf{E}_{x,y} &= \sum_{u,v} w_{u,v}[I(x+u, y+v) - I(x,y)]^2 \\ &= \sum_{u,v} w_{u,v}[xX + yY + O(x^2, y^2)]^2 \end{aligned}$$

where $w_{u,v}$ is a Gaussian function used to smooth the margins of the pixel regions to get less noisy gradients and X, Y are the gradients of I computed relatively on the directions u and v. For small shift, the gradient E can be written as

$$E(x, y) = Ax^2 + By^2 + 2Cxy$$

where

$$\begin{aligned} A &= X^2 \otimes w \\ B &= Y^2 \otimes w \\ C &= XY \otimes w \end{aligned}$$

Writing the equation for E in matrix form, the matrix M is defined as follows:

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

Let be α and β the two eigenvalues of the matrix M, the values of them is proportional to the magnitude of the gradient in the respective direction. Three different cases emerges:

- $\alpha \gg \beta$ or $\beta \gg \alpha$ denote the presence of an edge.
- $\alpha \sim \beta \sim 0$ denote a flat region
- $\alpha, \beta \ll 0$ denote the presence of a corner

Harris's corner detector is very fast to compute but suffers greatly from scale and rotation changes in the image. In a visual SLAM scenario it is required for a detector to be able to recognize the same features over intense orientation and scale change.
SIFT (Scale Invariant Feature Detector) from **Lowe** (2004) [10] represents a big step forward in the field of image processing and feature detecting. This algorithm (which belongs to the family of DoG, difference of Gaussians) employs a cascade filtering approach to select the most consistent set of feature from the image. Four major stages of computations are used: first a difference of Gaussians function is employed to identify potential interest points

that are invariant to scale and orientation, then a detailed model is fit to represent each candidate image region. One or more orientations are assigned to each keypoints base on local image gradient directions as to refer future operations to the same scale and orientation, providing invariance to these transformations. At the selected scale the image gradients are then computed and transformed into the final representation. The final feature properties are stored in a vector of dimension 128 referred to as the "descriptor", two similar features shares a similar descriptor.

The detector implemented in this thesis work is the SURF (Speeded-up Robust Features) detector by **Bay**, **Tuytelaars** and **Van Gool** (2006) [25]. SURF algorithm provides a robust scale and rotation invariant image feature descriptor in order to obtain great repeatability and consistency while mantaining a low computational cost, thus being more feasible in real-time applications. Second order deformations as skew, anisotropic scaling and perspective effects are not taken into account being their effect minor in relation to scaling and rotation. Being also the two cameras used in this thesis work optically calibrated, distortions are minimized, therefore the detector should behave well. SURF relies on the Fast-Hessian Detector, computing the follwing Hessian matrix for every point $\mathbf{x} = (x, y)$ in the image:

$$H(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{bmatrix}$$

where L_{xx} is the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$ with the image $I(x, y)$. Convolution of the Gaussian with the image is requested to minimize the noise: since the Gaussian acts like a low-pass filter, aliasing is reduced. When convoluted, the Gaussians are sampled so aliasing can still occur and second order derivatives should induce even more aliasing issues being their approximations in "box" shape ever more rough than the

standar Gaussian. This effect is proven to not being too influential therefore the increase in computational speed and efficiency is more relevant in this case. The maxima of the Hessian matrix determinant are chosen to be feature of interest in a pre-defined set of scales. To every feature detected now a descriptor is computed. In order to obtain rotation invariance, an orientation is assigned to every selected feature from the detector. The Haar-wavelet response is computed in x ad y directions on a circular neighbourhood around the feature of interest and weighted with a Gaussian. The wavelet responses are represented as vectors parallel to both axes and the orientation of the feature is a function of those two vectors.

The descriptor is then extracted following the next few steps: first a square region is constructed centered on the feature and oriented along the direction computed in the previous step. The region is then split in 4x4 square sub-regions and for reach subregions are sampled 5 equally spaced points. The Haar wavelet response is computed for every subregion on the sampled points and summed in both direction to generate the vector $\mathbf{v} = \{\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\}$. The full descriptor is just the union of this 4 element vector for each subregion resulting in a 64 element vector which is the normalized to obtain contrast invariance.

4.2.2 Stereo matching

When two images are acquired from a stereo camera setup the first step is feature detection. In this work the SURF detector is used because of the very close performances to SIFT [14] and because of the low computational time needed to perform a complete search for features. The two images are then compared to find the matching features that represent the projections of the same landmark in the two camera sensors. A matching feature must

satisfy two costraints:

- Feature descriptors must be similar. If a landmark is located at a sufficient distance from the cameras it should have very similar visual properties. This is not a robust requirement because the same appearance can be shared between a certain number of landmarks in the map, especially if the detector is invariant to scale and rotation. If identical objects are located in the field of view of the cameras (like desks, computers, etc) it is very likely that some visual features can not be distinguished from different objects.

- Epipolar costraint. For a couple of features detected in frame 1 and frame 2 to correctly represent the projections of the same landmark into the camera sensors, the epipolar costraint must be verified. Thus a match from two descriptors is rejected if it does not correctly satisfy a geometrical costraint.

Those two steps are respectively operated by evaluating the distance between feature descriptors and by a RANSAC filtering.

1. Descriptor match The first step is to evaluate the distance between all feature descriptors detected in frames 1 and 2. A SURF descriptor is a 1×64 vector, a 128×1 descriptor is also available but it is not employed in this work because the improvement in the feature description accuracy isn't needed. The distance can be evaluated as:

- SAD, *sum of absolute differences*. It is the sum of the absolute differences of every position in two feature descriptors.

$$SAD = \sum_{i=1}^{n} |f_1(i) - f_2(i)|$$

- SSD, *sum of squared differences*. It is the sum of the squared differences of every position in two feature descriptors.

$$SAD = \sum_{i=1}^{n} (f_1(i) - f_2(i))^2$$

For every feature descriptor in frame 1 the distance from all the descriptors in frame 2 is computed. The closest couple of descriptors is considered as a match if a threshold is satisfied: defining a numerical value for a maximum distance is a quick and straightforward approach even though it can be not a flexible approach because the distance from correct matches is very situation-dependent (changes in scales, rotation, illumination and effects of optical distortions induce numerical variations in the descriptor).

Algorithm 2 Stereo feature matching

$f_{cam1} \leftarrow \{f_1,\ f_2,\ ,\ \dots\ , f_n\}$ ▷ Descriptors from image 1
$f_{cam2} \leftarrow \{f_1,\ f_2,\ ,\ \dots\ , f_m\}$ ▷ Descriptors from image 2
for $i\ =\ 1\ :\ n$ **do**
  for $j\ =\ 1\ :\ m$ **do**
    $d_j \leftarrow distance(f_i - f_j)$ ▷ SAD or SSD distance
  end for
  $[i, j, d_j] \leftarrow min(d_j)$ ▷ index of features 1 and 2 with minimum distance
  if $d_j > thresh$ **then** reject match
  end if
end for

If the same descriptor is matched with more than one descriptors then the ambiguity can be solved rejecting the matches with higher distances.

2. RANSAC filtering In this step the matches obtained by association of the visual properties are filtered employing the epipolar costraint. Matches that do not satisfy the costraint are rejected since they do not represent the projection of the same landmark in the environment. Being the extrinsec parameters of the stereo setup known, the fundamental matrix F could be easily computed to test the model described in eq. 4.15 on all the feature pixel coordinates. Since the algorithm is not written to be implemented real-time, a high computational cost is not an issue. A more complex computation is performed: the RANSAC algorithm explained in alg. 1 is applied to estimate the fundamental matrix from a set of random inliers, testing it on all the matches and returning the most fitting model as well as the set of inliers and outliers. The outliers, matches that do not satisfy the costraint, are rejected. Since the fundamental matrix F, although 3×3, has only 7 degrees of freedom [20], it can be estimated from at least 7 equations given a set of 7 matching feature locations. The resulting system appears to be solvable by a lineear least squares regression. Being

$$F = \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix}$$

and being $x_1 = \{u_1,\ v_1,\ 1\}$ and $x_2 = \{u_2,\ v_2,\ 1\}$ eq. 4.15 can be written as

$$f_1u_1u_2 + f_2u_1v_2 + f_3u_1 + f_4v_1u_2 + f_5v_1v_2 + f_6v_1 + f_7u_2 + f_8v_2 + f_9 = 0$$

This equation is linear in f_i. Rather than ordinary least squares minimization, orthogonal least squares is employed. Let be $\mathbf{f} = \{f_1, f_2, ..., f_p\}$ a hyperplane to be fitted through a set of n points $\mathbf{z}_i = \{z_1, z_2, ..., z_p\}$ taking the centroid of the data as origin. Assuming that the noise on the data is Gaussian and that the elements of $\mathbf{z}$ have equal variance, the best fitting

hyperplane is obtained minimizing the perpendicular sum of Euclidean distances from the given data

$$min \sum_{i=1}^{n} (\mathbf{f}^T \mathbf{z}_i)^2$$

Let $\mathbf{Z}$ be the $n \times p$ measurement matrix with rows $\mathbf{z}_i$ and let be $\mathbf{M}$ the $p \times p$ moment matrix with eigenvalues λ_i and the correspondent eigenvectors $\mathbf{u}_i$ for $i = 1 : p$ forming an orthonormal system. The best fitting hyperplane is given by the eigenvector $\mathbf{u}_i$ corresponding to the minimum eigenvalue λ_i of the moment matrix. Before computing the moment matrix, the measurement matrix is centered subtracting to every row its mean value.

$$\begin{aligned} \mathbf{z} &= \{u_1u_2, u_1v_2, u_1, v_1u_2, v_1v_2, v_1, u_2, v_2, 1\}^T \\ \mathbf{Z} &= \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_p \end{bmatrix} \\ \mathbf{M} &= \mathbf{Z}^T\mathbf{Z} \end{aligned} \tag{4.16}$$

Now the eigenvalues of $\mathbf{M}$ are computed and the eigenvector correspondent to the minimum eigenvalue is $\mathbf{f}$ so the fundamental matrix is obtained. This procedure is summarized in alg. 3.

The full RANSAC algorithm to perform match filtering is very similar to alg. 1. The differences lie on the conditions to stop the iterations. The loop stops when *besterror* is lower than a threshold to ensure that no computational time is wasted, if then *besterror* is not low enough after a significant number of iterations the minimum consensus required is lowered and the iteration index is zeroed. The higher is the minimum consensus required to choose a

Algorithm 3 Orthogonal least squares estimation of the fundamental matrix

$l_1 \leftarrow \{u_1(1),\ v_1(1),\ 1; u_1(2),\ v_1(2),\ 1\ ; ...; u_1(n),\ v_1(n),\ 1\}$ ▷ matrix storing the pixel location of previously matched features

$l_2 \leftarrow \{u_2(1),\ v_2(1),\ 1; u_2(2),\ v_2(2),\ 1\ ; ...; u_2(n),\ v_2(n),\ 1\}$

for $i = 1 : n$ **do**

$Z(i,:) \leftarrow [u_1u_2(i), u_1v_2(i), u_1(i), v_1u_2(i), v_1v_2(i), v_1(i), u_2(i), v_2(i), 1]$

$Z(i,:) \leftarrow Z(i,:) - mean(Z(i,:))$ ▷ Centering

end for

$M = Z^T Z$

$[\lambda_1, \lambda_2, ..., \lambda_n], [u_1, u_2, ..., u_n] \leftarrow eig(M)$

$[u, \lambda] \leftarrow$ minimum eigenvalue and correlated eigenvector

$f \leftarrow u$

$$F \leftarrow \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix}$$

good model the lower is the chance to find a candidate. This can be explained taking into consideration, for instance, a linear regression where the dataset is a number of points scattered uniformly on a plane. If the threshold to consider a point as an inlier is low, it is very likely that a minimum consensus is never obtained.
In the algorithm, an error is computed as follows

$$\epsilon_i = \mathbf{x}_1(i)\ F\ \mathbf{x}_2(i)$$

since the result of the computation for a perfect match should be zero and *better error* is computed as the standard deviation of ϵ_i for $i = 1 : n$. Algorithm 4 summarize the whole RANSAC filtering procedure.

4.2.3 Data association

Data association is a crucial task in every SLAM environment: contrary to visual odometry algorithms where no information about the map is stored, in a SLAM implementation the knowledge of observed landmarks is improved over time. The more a landmark is observed the higher is the accuracy on his localization and the more significant is its contribution to improve the posterior knowledge over the robot path. In traditional implementations where range and bearing sensors are used (like RADAR, LiDAR, etc.) every landmark is distinguished from each other by its location in the map. In order to associate an observation to a known landmark a criteria based on geometric proximity is used computing the *mahalanobis distance* from a measurement to every component of the known map. If a stereo-camera is used as a sensor, *visual properties* of the measurements can be used for data association. The use of feature descriptors for association purposes is known for being much more robust than *maximum likelihood* estimators or

Algorithm 4 Full RANSAC algorithm for stereo match filtering

$k \leftarrow 0$ ▷ interation index

$d \leftarrow$ minimum consensus to accept a candidate F

$t \leftarrow$ threshold value to identify an inlier

$data \leftarrow$ initial set of n matching feature coordinates $\{l_1, l_2\}_i,\ i = 1 : n$

$best\ error \leftarrow$ Inf

while $best\ error\ >\ thresh$ **do**

  if $k > 1000$ **then**

    decrease d of a small amount

    $k \leftarrow 0$

  end if

  $inliers \leftarrow$ sample more than 7 matching features from $data$

  $outliers \leftarrow$ features in $data$ that are not inliers

  $F \leftarrow$ compute F using alg. 3

  for $(\mathbf{x}_{1,i}, \mathbf{x}_{2,i})\ \epsilon\ outliers$ **do**

    $\varepsilon_i \leftarrow \mathbf{x}_{1,i}^T F \mathbf{x}_{2,i}$

    if $\varepsilon_i\ <\ t$ **then**

      add $(\mathbf{x}_{1,i}, \mathbf{x}_{2,i})$ to $inliers$

    end if

  end for

  if length($inliers$) $>\ d$ **then**

    $F_{better} \leftarrow$ compute F using the new set of $inliers$

    $better\ error \leftarrow$ error computed using F_{better} on $inliers$

    if $better\ error < best\ error$ **then**

      $best\ error \leftarrow better\ error$

      $best\ inliers \leftarrow inliers$

    end if

  end if

  $k \leftarrow k + 1$

end while

mahalanobis distance computations on geometric properties of the map and in monocular approaches is the only method available since a single camera is a only-bearing sensor.

Mahalanobis distance Let be $\mathbf{z}$ a 3D measurement and $\hat{\mathbf{z}}_i$ the predicted measurement for the i-th landmark known from previous observations. The *innovation error* is the difference between an actual measurement and a predicted measurement

$$\nu_i = \mathbf{z} - \hat{\mathbf{z}}_i$$

Let also be $\mathbf{S}_i$ the *innovation covariance* computed as a function of the uncertainties associated to the i-th landmark state and to the current measurement, see eq. 2.10. The *mahalanobis distance* is computed as follows

$$\mathbf{D}_i = \sqrt{\nu_i^T \ \mathbf{S}_i^{-1} \ \nu_i}$$

After being computed for all the known landmarks, a candidate match for the current measurement is selected for the minimum value of $\mathbf{D}_i$. Being the *closest* landmark does not translate into being the *same* landmark so a threshold must be applied to discard a false association. A value for this threshold is usually defined empirically. The importance of considering the uncertainty in the process of finding an appropriate match is depicted in fig. 4.4.

Maximum likelihood estimator Mahalanobis distance is also a *negative log likelihood*. The classical solution to the data association problem is to choose an index i, the 'name' of a landmark, that maximizes the likelihood of the current sensor measurement

$$\hat{i} = \underset{i}{\operatorname{argmax}} \ p(z|i, s_t, u_t)$$

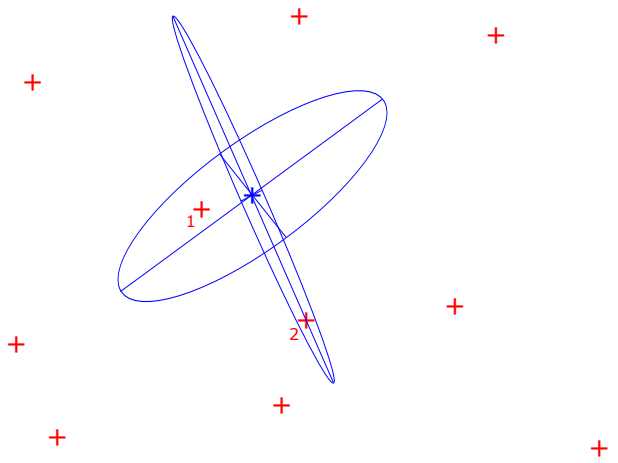


Figure 4.4: Effect of different shapes of the innovation covariance S: landmarks 1 and 2 (red crosses) are associated with the current measurement (blue cross) according to the innovation covariance which depends on the previous landmark uncertainty and current measurement uncertainty

As explained in detail in [15], the likelihood is calculated in closed form as follows

$$p(z|i, s_t, u_t) = \frac{1}{\sqrt{2\pi|S_i|}} e^{(-\frac{1}{2}\nu_i^T S^{-1}\nu_i)}$$

Visual data association Traditional data association based on nearest neighbour methods (like Mahalanobis distance minimization) can fail in many scenarios. If a dense set of landmarks is observed and the uncertainty associated to the measurement is sufficiently high, association can lead to ambiguous results being unable to distinguish landmarks very close to each other. Visual information about features detected in the image are uncorrelated with the geometry of the environment or with the pixel coordinates of the feature itself so even a very dense set of landmarks can be correctly recognized. In this case the arguments of the data association algorithm are the feature descriptors denoted by the letter $\mathbf{f}$, which are 1×64 vectors for SURF

detector. To every landmark in the map are associated its mean value and covariance for the state assuming that the distribution is Gaussian but also its visual descriptors $\{\mathbf{f}_{i,1}, \mathbf{f}_{i,2}, ..., \mathbf{f}_{i,n}\}$. Considering a landmark by just its appearance has its strength and weaknesses offering a very consistent association over little robot displacements but losing the ability to recognize it when it is observed from large viewpoint or scale changes. Previous results in the literature suggest that the consistency of the association is more important because, especially in EKF scenarios, poor associations can likely lead to divergence of the filter. Visual associations can be performed in two ways:

- *Euclidean distance* between descriptors. When a feature is returned by the detection algorithm, the association to a previously known feature is established by finding the minimum of

$$D_{Eucl} = \sqrt{(\mathbf{f} - \mathbf{f}_i)^T (\mathbf{f} - \mathbf{f}_i)}$$

  where i is the index of a landmark in the map. If D_{Eucl} is higher than a threshold empirically determined, the landmark is treated as a new observation and its descriptor is stored and given a new index.

- *Mahalanobis distance* between descriptors [21]. When multiple observation of the i-th landmark have been performed and every descriptor stored, a covariance matrix can be computed over those multiple observation of the same feature. To compute a covariance matrix the assumption of Gaussian distribution for the values of the descriptors is necessary. Just a few observations are not sufficient for obtaining a covariance matrix that is not ill-conditioned and that represent efficiently a Gaussian distribution. The covariance matrix is computed assuming that all the elements in the descriptor are indepent so the matrix is

diagonal. Then the following distance is computed

$$D_{Mahal} = \sqrt{(\mathbf{f} - \mathbf{f}_{i,avg})^T \; C_i^{-1} \; (\mathbf{f} - \mathbf{f}_{i,avg})}$$

where $\mathbf{f}_{i,avg}$ is the average descriptor associated to the i-th feature tracked along consequent time steps and C_i is the covariance matrix computed on those descriptors. Again, if D_{Mahal} is higher than a threshold the feature is marked as a new observation.

4.3 Stereo FastSLAM implementation

In this section the algorithm for the SLAM implementation using a stereo-camera as the only sensor is explained in detail. In the formulation for FastSLAM algorithm, data association is performed on a per-particle basis meaning that the association depends on the motion hypothesis for the robot at the current time steps. That makes absolute sense if the arguments on which data association is performed are the landmark poses in the known map because mainly of two reasons: the measurement model is a function of the robot pose at the current time step and for every particle an individual map is associated and corrected according to the obserations inherent to a particular pose hypotesis. In all the implementations for this thesis work, data association is independent of the robot pose because it relies only on visual properties of the features associated to the landmarks in the map. Thus a loop for data association can be performed once for every time step saving an computational time which is linear in the number of particles.

The **main loop** starts importing the two images correspondent to camera 1 and camera 2, then image features are computed employing SURF detector and a descriptor is associated to each feature. Stereo matching is then performed to find the matching features in the left and right images and a

RANSAC filtering procedure is employed for discarding erroneous matches which results in triangulations of non existing points.

Then **data association** is performed. For every matched feature in the two images the average descriptor is computed as

$$\mathbf{f}_j = \frac{\mathbf{f}_{j,left} + \mathbf{f}_{j,right}}{2}$$

and then a match for $\mathbf{f}_j$ is searched computing the Euclidean distance or the Mahalanobis distance from every other known descriptor associated to any landmark in the map. If a match is found with a distance lower than a threshold, an association for the j-th feature is found. A temporary array stores the associations from the current observations to the map and it will be used in the loop for landmark state update or initialization.

Triangulation of the observed landmarks is now performed. As stated before, the stereo-camera is used as a range and bearing sensor resulting as a *measurement* a fully initialized Euclidean location $\hat{\mathbf{x}}_j$ with its associated *measurement covariance* $\hat{\mathbf{P}}_j$. It is made the assumption that the distribution of the landmarks state is Gaussian.

Now the **particle loop** can start. The number of particles is choosen as an input from the user. While a dense particle set is ideal to correctly approximate the exact posterior over robot poses, the computational time is linear in the number of particles because for every particle taken into consideration a full map update is computed according to a single motion hypothesis. Previous work shows that a significant increase in localization performance is obtained from augmentation of the particle number from 10^0 to 10^2 but for more than 10^2 particles no significant improvement is achieved. For every

particle a motion hypothesis is sampled from a proposed distribution:

$$\begin{aligned} \mathbf{t}_{i,i-1}^{i-1} &\sim p(\mathbf{t}_{i,i-1}^{i-1}|\mathbf{s}_{i-1}, \mathbf{u}_i) \\ \phi^{i-1} &\sim p(\phi^{i-1}|\mathbf{s}_{i-1}, \mathbf{u}_i) \end{aligned} \tag{4.17}$$

where $\mathbf{t}_{i,i-1}^{i-1}$ is the displacement of the stereo-camera from step (i-1) to actual step (i) in the reference frame of the camera at step (i-1) and ϕ^{i-1} is the angle of rotation expressed in the reference frame at step (i-1). The motion model is represented as a multivariate Gaussian distribution of mean μ and covariance $\mathbf{Q}$ where

$$\begin{aligned} \mu &= \begin{bmatrix} t_z \\ t_x \\ \phi \end{bmatrix} \\ Q &= \begin{bmatrix} \sigma_z^2 & 0 & 0 \\ 0 & \sigma_x^2 & 0 \\ 0 & 0 & \sigma_\phi^2 \end{bmatrix} \end{aligned} \tag{4.18}$$

Then, according to the robot pose history, the total displacement and rotation expressed in the global reference frame is computed as follows in an iterative way:

$$\begin{aligned} \mathbf{t}_{i,1}^1 &= \mathbf{R}_{i-1}^1 \, \mathbf{t}_{i,i-1}^{i-1} + \mathbf{t}_{i-1,1}^1 \\ \mathbf{R}_i^1 &= \mathbf{R}_{i-1}^1 \, \mathbf{R}_i^{i-1} \end{aligned} \tag{4.19}$$

where the number 1 refers to the first reference frame as the global reference frame and $\mathbf{R}$ is the rotation matrix computed from ϕ. Recall that R_A^B is the matrix that rotates frame B to frame A. The total displacement and

rotation in the global reference from eq.4.20 will be used to relate current measurements in the local frame to the global frame and viceversa in the prediction step for the landmarks state.

Now all the triangulations or measurements are individually taken into account for landmark **initialization** or **update**. If the j-th observation is related to a new landmark, its coordinates and covariance are stored in a new position in the map with its feature descriptor. A transformation from local frame (i) to global frame (1) is performed. Being $\hat{\mathbf{x}}_j^i$ and $\hat{\mathbf{P}}_j^i$ the Euclidean coordinates and covariance for landmark j in the reference frame i,

$$\begin{aligned} \hat{\mathbf{x}}_j^1 &= \mathbf{R}_i^1 \, \hat{\mathbf{x}}_j^i + \mathbf{t}_{i,1}^1 \\ \hat{\mathbf{P}}_j^1 &= \mathbf{R}_i^1 \, \hat{\mathbf{P}}_j^i \, \mathbf{R}_i^{1T} \end{aligned} \tag{4.20}$$

If j-th measurement is associated to a landmark in the map, an EKF update is performed. First, the latest information about state and covariance of this landmark are recalled and the **prediction step** is performed.

Prediction step Let be $\mathbf{x}_{j,last}^{1,[m]}$ and $\mathbf{P}_{j,last}^{1,[m]}$ the last known state and covariance for the landmark j. The superscript [m] denotes that the state is related to the m-th particle, for all the other particles the state will differ according to the corrections made from slightly different poses. According to the robot path fot the m-th particle, the *predicted measurement* is obtained just referring landmark j to the current reference frame. This represents the *prior knowledge* over the landmark state, that is the knowledge before any actual

measurement is taken into account.

$$\begin{aligned} \mathbf{x}_{j,(-)}^{i,[m]} &= \mathbf{R}_1^{i,[m]}\ \mathbf{x}_{j,last}^{1,[m]} + \mathbf{t}_{i,1}^{i,[m]} \\ \mathbf{P}_{j,(-)}^{i,[m]} &= \mathbf{R}_1^{i,[m]}\ \mathbf{P}_{j,last}^{1,[m]}\ \mathbf{R}_1^{i,[m]T} \end{aligned} \tag{4.21}$$

Update step The state of the landmark is now updated correcting the *predicted measurement* with the actual measurement. Because of the equivalence of *state* and *measurement* in this implementation, the *measurement model* which is the function that returns the measurement known the state of a landmark is trivial.

$$\hat{\mathbf{x}}_j = h(\hat{\mathbf{x}}_j \mid \mathbf{s}_i^{[m]}, \mathbf{x}_j^{[m]}) = [1]\ \mathbf{x}_j^{i,[m]}$$

Therefore both the prediction model, which is a rototraslation, and the measurement model, which is an identity, are linear. The extended Kalman filter in this particular formulation is just a Kalman filter. This way errors induced by non-linearities are absent.
The *innovation error* is computed as

$$\nu_j = \hat{\mathbf{x}}_j - \mathbf{x}_{j,(-)}^{i,[m]}$$

Then the *innovation covariance* S_j and the Kalman gain are computed as

$$\begin{aligned} \mathbf{S}_j^{[m]} &= \mathbf{P}_{j,(-)}^{i,[m]} + \hat{\mathbf{P}}_j \\ \mathbf{K}_j^{[m]} &= \mathbf{P}_{j,(-)}^{i,[m]}\ \mathbf{S}_j^{[m]-1} \end{aligned} \tag{4.22}$$

The Jacobian of the measurement function is omitted as it is the identity matrix. Then the state of landmark j is updated as follows:

$$\mathbf{x}_{j,(+)}^{i,[m]} = \mathbf{x}_{j,(-)}^{i,[m]} + \mathbf{K}_j^{[m]}\ \nu_j$$

Weigth The contribution of a landmark's state update to the weight of particle m is computed as follows

$$\mathbf{w}_j^{[m]} = \frac{1}{\sqrt{2\pi|\mathbf{S}_i|}} e^{(-\frac{1}{2}\nu_i^T \mathbf{S}^{-1}\nu_i)}$$

The weight in FastSLAM scenario should represent the likelihood of the measurements, the more a particle is close to the actual location of the robot, the more likely are the observations obtained. The set of measurements should be a set of independent observations therefore if

$$\mathbf{w}^{[m]} \sim p(\mathbf{z}|\mathbf{s}_t, \mathbf{u}_t)$$

by marginalizing the posterior distribution of the measure $\mathbf{z}$

$$\begin{aligned} \mathbf{w}^{[m]} &\sim p(\mathbf{z}_1)\ p(\mathbf{z}_1)\ ...\ p(\mathbf{z}_n) \\ &= \prod_{j=1}^{n} \mathbf{w}_j^{[m]} \end{aligned} \tag{4.23}$$

An empirical approach is to compute the total weight of the particle by a sum of all the contributions because being the value of $\mathbf{w}_j^{[m]}$ generally low, computing a very dense map the total weight can lead to arithmetic underflow, so

$$\mathbf{w}^{[m]} = \sum_{j=1}^{n} \mathbf{w}_j^{[m]}$$

Resampling When all the particles have been evaluated and their weight computed, resampling takes place. To resample means picking from the old particle set a new set of M particles where the likelihood for a particle to be sampled is proportional to its weight. Resampling is performed "with replacement", that means that a single particle can be picked more than once.

4.4 Mono FastSLAM implementation

While a stereo setup is a solution for range and bearing measurements, monocular estimations are only capable of returning bidimensional information about the environment as only a *projection* of the map is returned by the camera. A single camera is therefore a bearing-only sensor. The main challenge in mapping algorithms using monocular estimations is the reconstruction of 3D properties of an observed environment from 2D measurements. In this thesis two different approaches are formulated, the main difference is the way depth information are obtained from consequent observation of image features correspondent to landmarks in the map. The first approach is to obtain depth information from an EKF update initializing a landmark imposing a very weak prior on the first acquisition and then updating the Euclidean position in the map from bidimensional measurements as the pixel coordinates in the image. When the uncertainty over a landmark locations is sufficiently low, the landmark can contribute to improve pose estimation. The other approach is to acquire depth information by triangulation over consequent time steps. The history of measurements for every landmark is stored in a buffer and every time a landmark is observed the parallax over the images on the buffer is computed. When the parallax is above a threshold depth is known by triangulation. Every following triangulation of a landmark is used to update the state via a Kalman filter update similarly to the stereo implementation.

4.4.1 Visual Monocular SLAM

The procedure for image acquisition and feature detection is identical to the stereo algorithm. This time only the images from one of the two cameras

from the same dataset are stored, this way the computational time decreases significantly. Data association is performed as explained in section 4.2.3, the only difference is that no average descriptor is computed because only one feature descriptor per step is available. An interesting approach would be computing an average descriptor over the multiple detections for every landmark to extend the scale and viewpoint changes from which a feature can be associated. Since the optical distortions of the images captured from the cameras are corrected, feature tracking is precise and consistent over relatively long distances so data association is performed well even in the most straightforward approach. In the following chapter the advantages of different techniques in data association will be investigated. Data association is performed outside the particle loop because of the pose-invariancy of the visual associations.

The **particle loop** is articulated as in the stereo algorithm, camera motion is modeled as in eq 4.18. In this work it is evaluated just the motion of the left camera even in the stereo implementation as to perform the closest comparison between the two algorithms.

Landmark initialization If a feature is seen for the first time, its image descriptor is stored and a new index is assigned to the landmark. In order for its Euclidean coordinates to be updated in the following observations, a position in the map has to be initialized. In the main FastSLAM algorithm, in order to initialize a landmark, the measurement function must be inverted (if feasible). That means obtaining 3D information from a 2D measurement which is, of course, a problem of infinite solutions since every landmark location that lie on a line from the camera optical center to the projection on the image plane gives the same 2D measurement.

To overcome this issue, which is intrinsec in every monocular mapping problem, a very weak prior about the landmark depth is assigned. This initial distribution represent approximately the fact that the landmark must lie in front of the camera. A default value for the depth as with a very high uncertainty is assigned to the new landmark, then the *projection function* is inverted and a full 3D location is known. The resulting 3D covariance can be represented as a tight ellipsoid with two reasonably small and even semiaxis and the major semiaxis much bigger and parallel to the preimage line of the landmark. This permits the depth to be corrected in the next steps while "locking" the landmark in the near region of the first preimage line. The length of the small semiaxes is a function of the generally low uncertainties over the intrinsic parameters. It should be noted that a single stereo measurement returns a wider ellipsoid because it takes into account not only the uncertainties on the extrinsic parameters but also the uncertainties over the intrinsic parameters of both the cameras to which the triangulation error depends.

Let be $\mathbf{z}_j = \{u, v\}_j^T$ the *measure* for landmark j and $\mathbf{x}_j = \{x, y, z\}_j^T$ its Euclidean state. The *measurement function* or *projection function*, which is the function that returns the projection on the image plane in pixel coordinates of a landmark state in Euclidean coordinates, is modeled as follows:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} S_x f & 0 & u_0 \\ 0 & S_y f & v_0 \end{bmatrix} \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{bmatrix} \tag{4.24}$$

Its Jacobian is therefore:

$$J = \begin{bmatrix} \frac{\delta u}{\delta x} & \frac{\delta u}{\delta y} & \frac{\delta u}{\delta z} \\ \frac{\delta v}{\delta x} & \frac{\delta v}{\delta y} & \frac{\delta v}{\delta z} \end{bmatrix} = \begin{bmatrix} \frac{S_{xf}}{z} & 0 & -\frac{S_{xf} x}{z^2} \\ 0 & \frac{S_{yf}}{z} & -\frac{S_{yf} y}{z^2} \end{bmatrix} \tag{4.25}$$

Eq 4.24 must be inverted to obtain a 3D location for landmark j when measure

$\mathbf{z}_j = \{u, v\}_j^T$ is known. The first step is to assign a default value for z_j that can take into consideration the environment in which the camera is moving. In this particular case a value of 5000 mm is chosen because it is a likely depth for landmarks in a room of approximate length of 10 meters. By fixing $\lambda = z_j$ eq 4.24 can be written in homogeneous coordinates as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} S_x f & 0 & u_0 \\ 0 & S_y f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{x}{\lambda} \\ \frac{y}{\lambda} \\ 1 \end{bmatrix}$$

The *inverse projection* function is then written as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \lambda \begin{bmatrix} S_x f & 0 & u_0 \\ 0 & S_y f & v_0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{4.26}$$

The covariance $\mathbf{P}_j$ is computed by propagating the uncertainty on intrinsic parameters as well as the user-assigned uncertainty on λ.

Landmark update If a landmark is already a part of the known map, an EKF update is performed. Contrary to the stereo implementation, where the update involves the computations of linear functions, this time the update is performed using an actual extended Kalman filter because of the non-linearities that occur evaluating the measurement model (eq 4.24 and 4.25) where the source of non-linearity is the normalization via the coordinate z. First the last state and covariance $\mathbf{x}_{j,last}^{[m]}$ $\mathbf{P}_{j,last}^{[m]}$ for landmark j are recalled. In order to compute an *expected measurement* for this landmark, a change of reference frame from the global reference to the local is necessary. This operation is performed on a *per-particle basis* so the path is known without uncertainty. This allows the update to not take into consideration the uncertainty on the robot pose to perform the reference change.

Computing $\mathbf{R}_1^i$ and $\mathbf{t}_{1,i}^i$ (the superscript [m] is omitted. Translations and rotations are intrinsic to every particle involved in the computation.) the values for $\mathbf{x}_{j,last}^{i,[m]}$ and $\mathbf{P}_{j,last}^{i,[m]}$ are obtained. By the means of equations 4.24 and 4.25 the *expected measurement* is computed as the projection on the image plane in pixel coordinates that landmark j should return according to the *prior* knowledge of its state.

$$expected = h(\mathbf{x}_{j,last}^{i,[m]})$$

Recalling the image coordinates returned from the detector for the image feature correspondent to landmark j, the *innovation error* is computed as:

$$\nu_j = \mathbf{z}_j - expected$$

Next, the Jacobian of the measurement model is computed over the current landmark state. The Jacobian is then used to compute the innovation covariance and Kalman gain to perform the update.

$$\begin{aligned} \mathbf{J}_j &= projectJac(\mathbf{x}_{j,last}^{i,[m]}, camera\ param) \\ \mathbf{S}_j &= \mathbf{J}_j^T\ \mathbf{P}_{j,last}^{i,[m]}\ \mathbf{J}_j + \mathbf{R}_j \\ \mathbf{K}_j &= \mathbf{P}_{j,last}^{i,[m]}\ \mathbf{J}_j\ \mathbf{S}_j^{-1} \end{aligned} \tag{4.27}$$

Where $\mathbf{R}_j$ is the estimated measurement covariance. $\mathbf{R}_j$ is a 2×2 matrix where the dimensions are related to the pixel coordinates u and v.

$$\mathbf{R}_j = \begin{bmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix}$$

The state update is the performed as follows:

$$\begin{aligned} \mathbf{x}_{j,(+)}^{i,[m]} &= \mathbf{x}_{j,last}^{i,[m]} + \mathbf{K}_j\ \nu_j \\ \mathbf{P}_{j,(+)}^{i,[m]} &= ([\mathbf{I}] - \mathbf{K}_j\ \mathbf{J}_j^T)\ \mathbf{P}_{j,last}^{i,[m]} \end{aligned} \tag{4.28}$$

Where the symbol $(-)$ denotes the prior knowledge over the state and covariance.

4.4.2 Triangulation Based Visual Monocular SLAM

This approach in tridimensional feature estimations from bidimensional measurements involves obtaining depth information by triangulation over following time steps. In section 4.1.2 an algorithm for triangulating landmark locations from projections on image planes of the cameras is explained. The same algorithm can be implemented on different images captured by the same camera. The baseline from the two optical centers will be the rototranslation from the two camera position therefore being heavily biased on the particles involved. While in the stereo environment and in the vision based monocular algorithm measurements are not dependent to the pose hypotesis being respectively triangulations from a fixed baseline and image feature detections, in this algorithm becase the baseline is dependent on the pose history evaluated, the maps inherent in different particles will differ substantially.
When a feature is seen for the first time its descriptor is stored and no computation is performed. In the following observations the parallax from the first observation to the current one is computed and when sufficiently high triangulation is performed. The parallax is the angle between two lines of sight of a landmark. Being the preimage lines known by the feature locations in the sensor, parallax is easily computed as follows.

Parallax computation Lets consider two camera poses denoted as 1 and 2 where a particular landmark is observed. The two optical centers are connected by vector $\mathbf{t}_{o1,o2}$ connects. It is made the assumption that the preimage lines intersect on the landmark P so the points $\{o_1, P, o_2\}$ are coplanar and

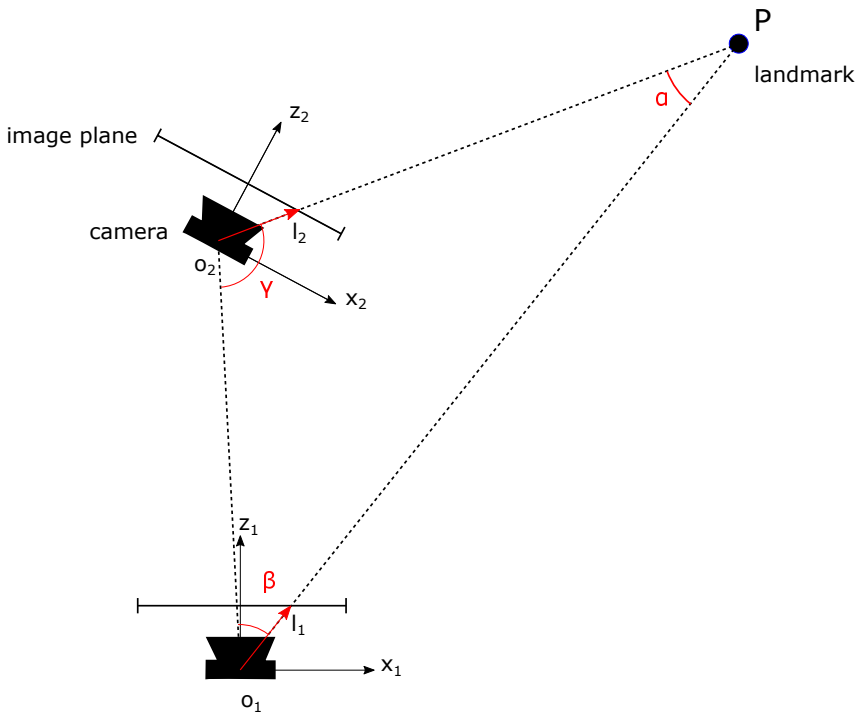


Figure 4.5: Triangle o_1 o_2 P. The preimage lines with its own directional vector are highlighted with dashed lines. α, β, γ are the inner angles of this triangle, α is the *parallax*. Vectors l_1 and l_2 connect the camera optical center to each feature location in metric units. It is made the assumption that l_1 and l_2 are coplanar.

define a triangle. Let be α, β, γ the inner angles of this triangle where α denotes the *parallax*. The following parameters are known: $\mathbf{l}_1^1$, $\mathbf{l}_2^2$, $\mathbf{t}_{1,o1}^1$, $\mathbf{t}_{1,o2}^1$, $\mathbf{R}_{o1}^1$, $\mathbf{R}_{o2}^1$ which are respectively the directional vector of the preimage lines in each reference frame, the translation and rotation from the initial position of the camera to poses 1 and 2. α can be computed from

$$\alpha + \beta + \gamma = \pi$$

But for β and γ the following relations can be written:

$$\begin{aligned} \cos(\beta) &= \frac{\mathbf{l}_1 \cdot \mathbf{t}_{o1,o2}}{|\mathbf{l}_1| \cdot |\mathbf{t}_{o1,o2}|} \\ \cos(\gamma) &= \frac{\mathbf{l}_2 \cdot \mathbf{t}_{o1,o2}}{|\mathbf{l}_2| \cdot |\mathbf{t}_{o1,o2}|} \end{aligned} \tag{4.29}$$

Therefore, inverting the equations

$$\begin{aligned} \beta &= \arccos(\frac{\mathbf{l}_1 \cdot \mathbf{t}_{o1,o2}}{|\mathbf{l}_1| \cdot |\mathbf{t}_{o1,o2}|}) \\ \gamma &= \arccos(\frac{\mathbf{l}_2 \cdot \mathbf{t}_{o1,o2}}{|\mathbf{l}_2| \cdot |\mathbf{t}_{o1,o2}|}) \end{aligned} \tag{4.30}$$

The vectors involved in the computation are referred to the global reference frame by the means of the following equations:

$$\begin{aligned} \mathbf{l}_1 &= \mathbf{R}_{o1}^1 \, \mathbf{l}_1^1 \\ \mathbf{l}_2 &= \mathbf{R}_{o2}^1 \, \mathbf{l}_2^2 \\ \mathbf{t}_{12}^1 &= \mathbf{t}_{1,o2}^1 - \mathbf{t}_{1,o1}^1 \end{aligned} \tag{4.31}$$

Then α is known from

$$\alpha = \pi - \beta - \gamma$$

Landmark update When the parallax is above a threshold, landmark j is triangulated. If it is the first time being triangulated then its state and covariance are stored and the next landmark is evaluated. If otherwise a triangulation has been previously performed, an Extended Kalman update takes place by the same procedure explained in section 4.4.1.

4.5 Visual Odometry

Estimating camera rotations and displacements from consecutive images is an efficient way to perform odometry on a moving robot. Differently from dead reckoning (integration of accelerations over time), visual odometry can be performed on large robot displacements to reduce drifts and it is much more reliable than wheel odometry. Visual odometry is actively performed [13] on the martian rovers to correct odometry information and to improve the ego-motion estimation through difficut manouvers. Visual odometry is still subject to drifts over time, especially if image features are tracked with low parallax changes, so it is a viable option to correct visual odometry information using a SLAM algorithm.

Stereo 3D-to-3D Visual Odometry After the feature points from left and right images over two consecutive time steps are extracted, feature matching takes place to find correspondences. It is necessary that every feature involved in the computation is found in all 4 images therefore the relative motion of the Euclidean parametrized landmark can be evaluated. The relationship between features is depicted in fig. 4.6. Let ${}^{P1}\mathbf{X}$ denote that the point is calculated with reference to the first camera when the stereo setup is in the initial position P1. After the rototranslation from position P1 to position P2 all the points are triangulated again. ${}^{P2}\mathbf{X}$ refers to the

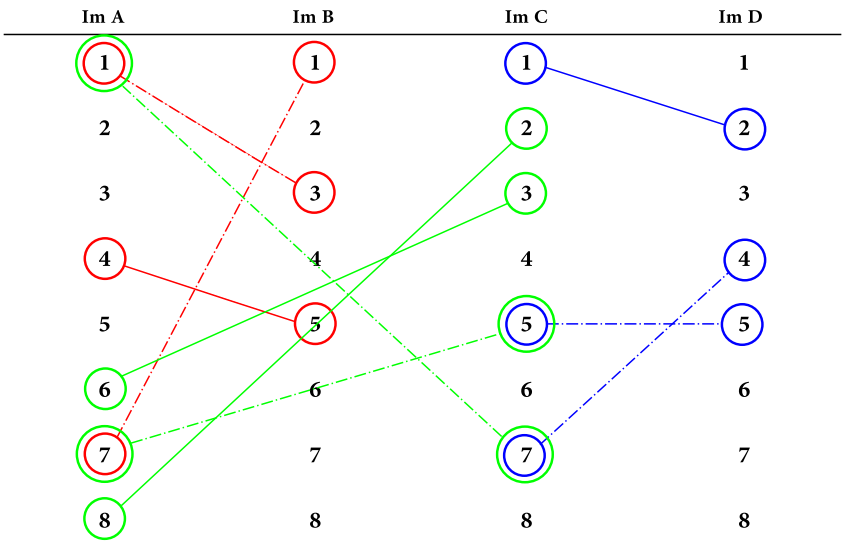


Figure 4.6: Correlation of the image features on the stereo visual odometry algorithm. im1 and im2 are the left and right images captured at time step t, im3 and im4 are the left and right images captured at time step t+1. After detecting all the features, correlations between all the 4 images have to be found (dashed lines).

point location when the setup is in position P2. The following equation can therefore be written:

$$^{P1}\mathbf{X}_i = {}^{P1}_{P2}\mathbf{R}\,{}^{P2}\mathbf{X}_i + {}^{P1}\mathbf{P}_{P2,P1}$$

Where ${}^{P1}_{P2}\mathbf{R}$ is the rotation matrix from frame 1 in the second position P2 to frame 1 in the first position P1; ${}^{P1}\mathbf{P}_{P2,P1}$ is the origin of frame 1 in P2 with reference to the origin of frame 1 in P1 and expressed in P1. The rotation matrix and translation vector are the information that have to be retrieved to evaluated the motion.

A linear least squares approach is used to obtain the camera motion. Let be

$$\mathbf{e}_i = {}^{P1}\mathbf{X}_i - {}^{P2}_{P1}\mathbf{R}\,{}^{P2}\mathbf{X}_i - {}^{P1}\mathbf{P}_{P2,P1}$$

the error that is a function of the unknown motion parameters. The cost function E is the function that has to be minimized

$$E = \sum_{i=1}^{n} |\mathbf{e}_i|^2$$

In order to separate the evaluation of rotation and translation, the centers of the two point clouds are defined:

$$^{P1}\mathbf{X} = \sum_{i=1}^{n} {}^{P1}\mathbf{X}_i$$

$$^{P2}\mathbf{X} = \sum_{i=1}^{n} {}^{P2}\mathbf{X}_i$$

And two new sets of 3D points are defined by:

$$^{P1}\mathbf{X}' = {}^{P1}\mathbf{X}_i - {}^{P1}\mathbf{X}$$

$$^{P2}\mathbf{X}' = {}^{P2}\mathbf{X}_i - {}^{P2}\mathbf{X}$$

It can be demonstrated that the cost function E is minimized when the cost function E' is minimized.

$$E = \sum_{i=1}^{n} |{}^{P1}\mathbf{X}'_i - {}^{P2}_{P1}\mathbf{R}\,{}^{P2}\mathbf{X}'_i|^2$$

Then, once the rotation has been evaluated, the following equation can be used to obtain the translation

$${}^{P1}\mathbf{P}_{P1,P2} = {}^{P1}\mathbf{X} - {}^{P2}_{P1}\mathbf{R}\ {}^{P2}\mathbf{X}$$

E' can be written as follows:

$$E' = \sum_{i=1}^{n} |{}^{P1}\mathbf{X}'_i|^2 + \sum_{i=1}^{n} |{}^{P2}\mathbf{X}'_i|^2 - 2\sum_{i=1}^{n}[\left({}^{P1}\mathbf{X}'_i\right)^T \left({}^{P2}_{P1}\mathbf{R}\,{}^{P2}\mathbf{X}'_i\right)]^2$$

which is minimum when the third term is maximum. Since

$$\mathbf{a}^T\mathbf{R}\mathbf{b} = Tr(\mathbf{R}^T\mathbf{a}\mathbf{b})$$

$$\begin{aligned}\sum_{i=1}^{n}[\left({}^{P1}\mathbf{X}'_i\right)^T \left({}^{P2}_{P1}\mathbf{R}\,{}^{P2}\mathbf{X}'_i\right)]^2 &= Tr({}^{P2}_{P1}\mathbf{R}^T\mathbf{M}) \\ \mathbf{M} &= \sum_{i=1}^{n}[\left({}^{P1}\mathbf{X}'_i\right)\left({}^{P2}\mathbf{X}'_i\right)^T]^2\end{aligned}$$

The rotation matrix can then be computed as follows:

$${}^{P2}_{P1}\mathbf{R} = \mathbf{M}(\mathbf{M}^T\mathbf{M})^{\frac{1}{2}}$$

The matrix $\mathbf{M}^T\mathbf{M}$ is positive definite thus its eigenvalues are positive and it can be expressed by the means of the following equation

$${}^{P2}_{P1}\mathbf{R} = \mathbf{M}(\sum_{i=1}^{3}\frac{1}{\sqrt{\lambda_i}}\hat{\mathbf{u}}_i\hat{\mathbf{u}}_i^T)$$

Chapter 5

Results and Analysis

	Camera 1 (left)	Camera 2 (right)	
$S_x f$	1133.20108	1136.43126	$[\frac{pix}{mm}]$
$S_y f$	1133.19673	1135.97654	$[\frac{pix}{mm}]$
x_0	1058.25306	1056.95157	[pix]
y_0	524.70888	542.00913	[pix]
f	6	6	[mm]
FOV	86x53	86x53	[°]

Table 5.1: Intrinsec parameters for the two cameras

The cameras employed in this work capture images at the resolution of 2040x1086 pixels, two identical wide angle lenses are used. Cameras have been calibrated via the Camera Calibration Toolbox[1], the intrinsec parameters as well as the lens specifications are summarized in the table 5.1. The stereo setup baseline is approximately 0.5 m and axes z_1 and z_2 are almost parallel, figure 5.1 shows the experimental setup.

[1] `http://www.vision.caltech.edu/bouguetj/calib_doc/`

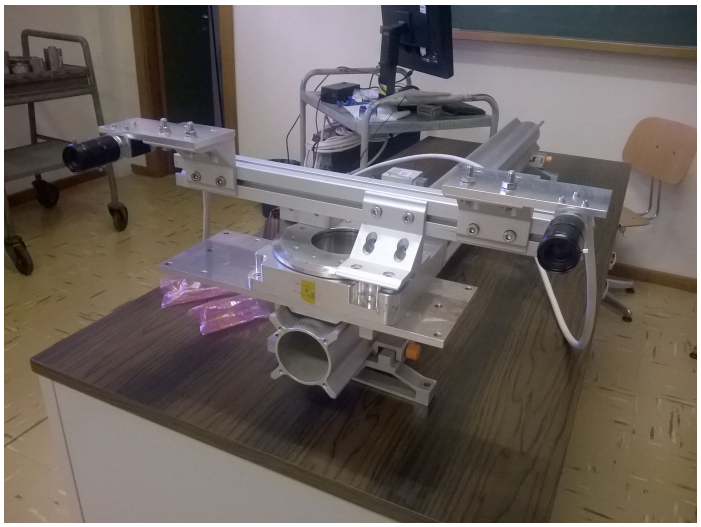


Figure 5.1: Picture of the stereo-camera setup

The algoritmhs for mono and stereo visual SLAM are first tested on the same dataset containing pictures obtained on a translational motion of 1350mm. The displacements of the stereo setup is approximately parallel to the z-axis of the two cameras. Since no odometry information is available, the proposed distribution of the camera pose and rotation is a multivariate Gaussian distribution centered on the expected camera position and orientation with an extremely high noise. This way an in-depth analisys of the ability of the two algorithms to reject bad motion hypotesys is performed. The proposed distribution in the stereo scenario is:

$$\mathcal{N}\left(\begin{bmatrix}0\\ \Delta z\\ \theta\end{bmatrix}, \begin{bmatrix}\sigma_x & 0 & 0\\ 0 & \sigma_z & 0\\ 0 & 0 & \sigma_\theta\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}0\ mm\\ 50\ mm\\ 0^\circ\end{bmatrix}, \begin{bmatrix}50/3\ mm & 0 & 0\\ 0 & 50/3\ mm & 0\\ 0 & 0 & 1^\circ\end{bmatrix}\right)$$

The frames employed for this analisys are captured approximately every 50mm so the noise imposed in the distribution involves sampling of null motion hypotesys and displacements equals to double the real translation. The noise added to the orientation is still very high causing to sample con-

secutive rotations that spans between $-3°$ and $3°$. A lesser ability to discard bad motion hypotesys of the monocular algorithm required the motion to be modeled with lower noise, so the proposed distribution employed for the monocular scenario is:

$$\mathcal{N}\left(\begin{bmatrix}0\\ \Delta z\\ \theta\end{bmatrix}, \begin{bmatrix}\sigma_x & 0 & 0\\ 0 & \sigma_z & 0\\ 0 & 0 & \sigma_\theta\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}0\ mm\\ 100\ mm\\ 0°\end{bmatrix}, \begin{bmatrix}20\ mm & 0 & 0\\ 0 & 20\ mm & 0\\ 0 & 0 & 0.5°\end{bmatrix}\right)$$

For the monocular algorithm frames are captured every 100mm translational steps. The ability of the two algorithms to correctly compute the real path and orientation of the stereo setup are investigated in relation to the number of particles employed. While the computational effort increases almost linearly to the number of particles, the accuracy in modeling the most likely path is proportional to the particle density. In the following figures, the sparsification of the motion hypotesis which occurs without resampling is plotted as a reference to highlight the ability of both the mono and stereo algorithms to costrain the motion near the true path. Then maps computed are then compared to highlight some of the strengths and issues related to both scenarios and to verify that the environment is correctly mapped resembling the actual geometry of the map. SURF descriptor and detector is employed with a threshold of 1000 which is low enough to detect from 150 to 250 features per frame. While in the monocular implementation every feature is taken into consideration for evaluating the parallax between frames and eventually triangulating the landmark, in the stereo scenario features are first matched between the left and right frames using their visual descriptors and then a RANSAC refinement takes place to eliminate matches that do not verify the epipolar costraint.

Stereo SLAM - Translational test The following results show the family of particles that survived the resampling after the last frame. It is implemented the strategy of performing the resampling step imposing a threshold on the weigth trading particle sparsification for path estimation accuracy. Particle sparsification is a prerequisite for loop closing procedures since particle filters based SLAM algorithms perform loop closing sampling from a given set of particles. If the robot returns to a previously known location, the particles (or trajectory hypothesis) that returns the lowest innovation errors are resampled while the others are discarded. If no particle intersects the previous robot location because the sparsification didn't extend as much as needed then loop closure is not obtained. If though loop closure is not expected to happen, a higher particle density can be obtained resampling only particles that have a minimum importance weight to discard bad motion hypothesis so that the following pose estimate can be more accurate. The results are obtained for no minimum weigth, 0.5 (normalized) minimum weigth and 0.8 minimum weigth. It is clearly evident that when no minimum weigth is set for resampling purposes, the sparsification of the particles remains relatively high and can more easily lead to a slow divergence of the computed path from the ground truth. As the threshold is set higher, particles are denser and closer to the actual path of the robot. The table 5.2 summarizes the best results for the stereo implementation.

	case 1 ($w_{min} = 0$)	case 2 ($w_{min} = 0.5$)	case 3 ($w_{min} = 0.8$)
particles	5000	1000	5000
ε_s [mm]	24.27 (1.80%)	17.38 (1.29%)	**2.62** (**0.19** %)
ε_θ [°]	0.167	0.022	**0.02**

Table 5.2: Numerical examples of three best scenarios

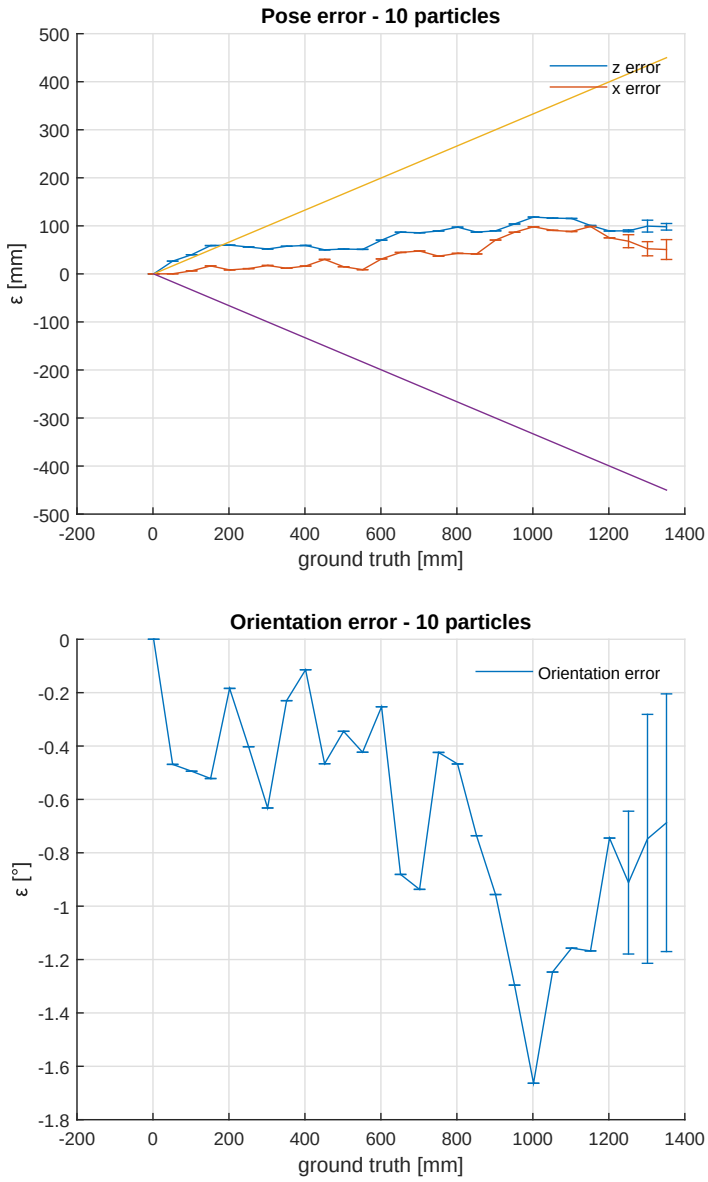


Table 5.3: Particles after the last resampling. Pose and orientation error vs imposed displacement. Oblique lines for odometry error bounds at 1 sigma.

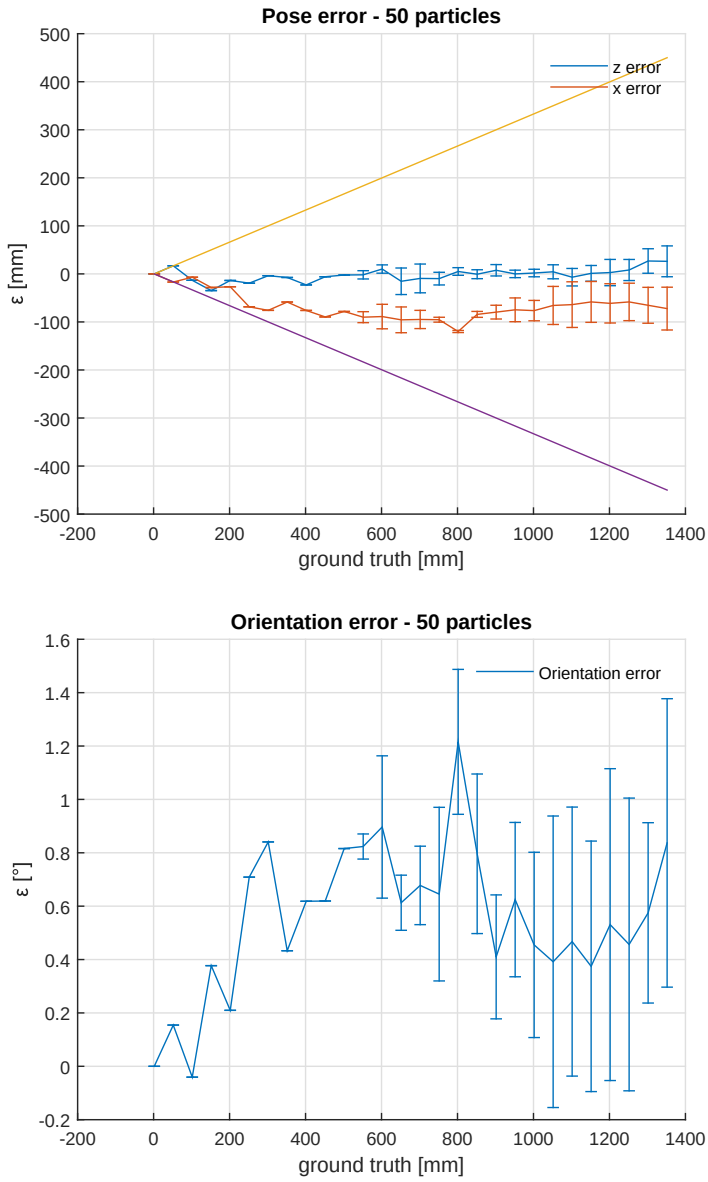


Table 5.4: Particles after the last resampling. Pose and orientation error vs imposed displacement. Oblique lines for odometry error bounds at 1 sigma.

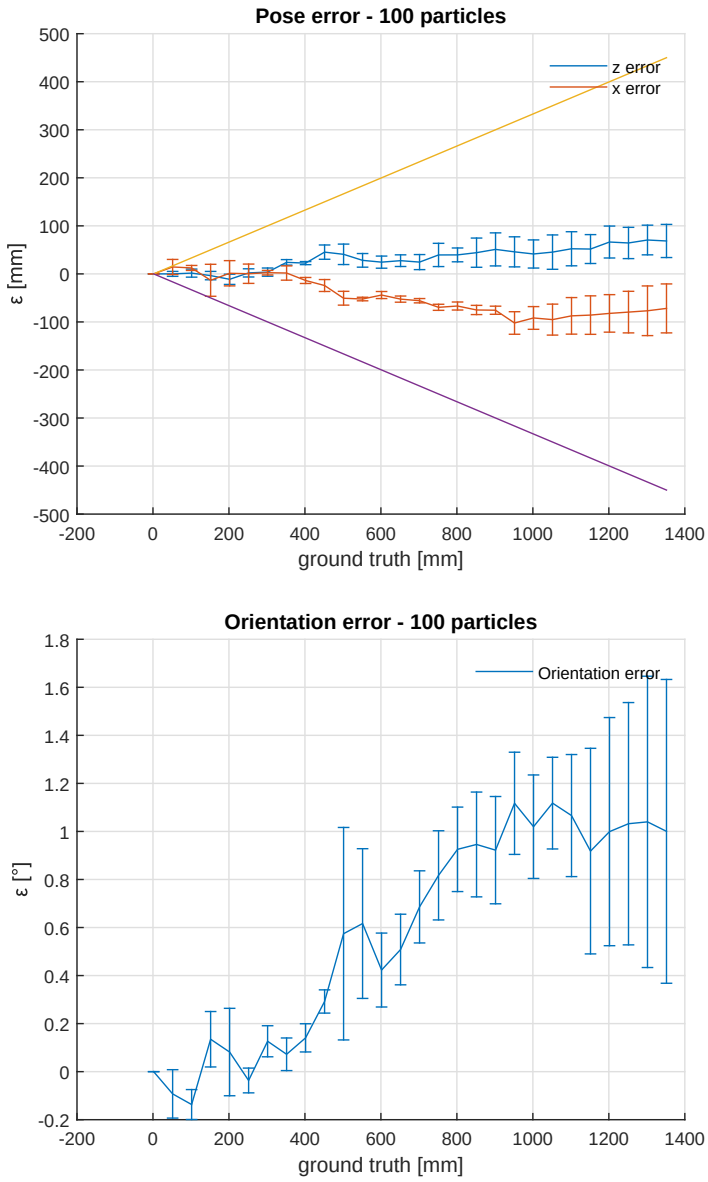


Table 5.5: Particles after the last resampling. Pose and orientation error vs imposed displacement. Oblique lines for odometry error bounds at 1 sigma.

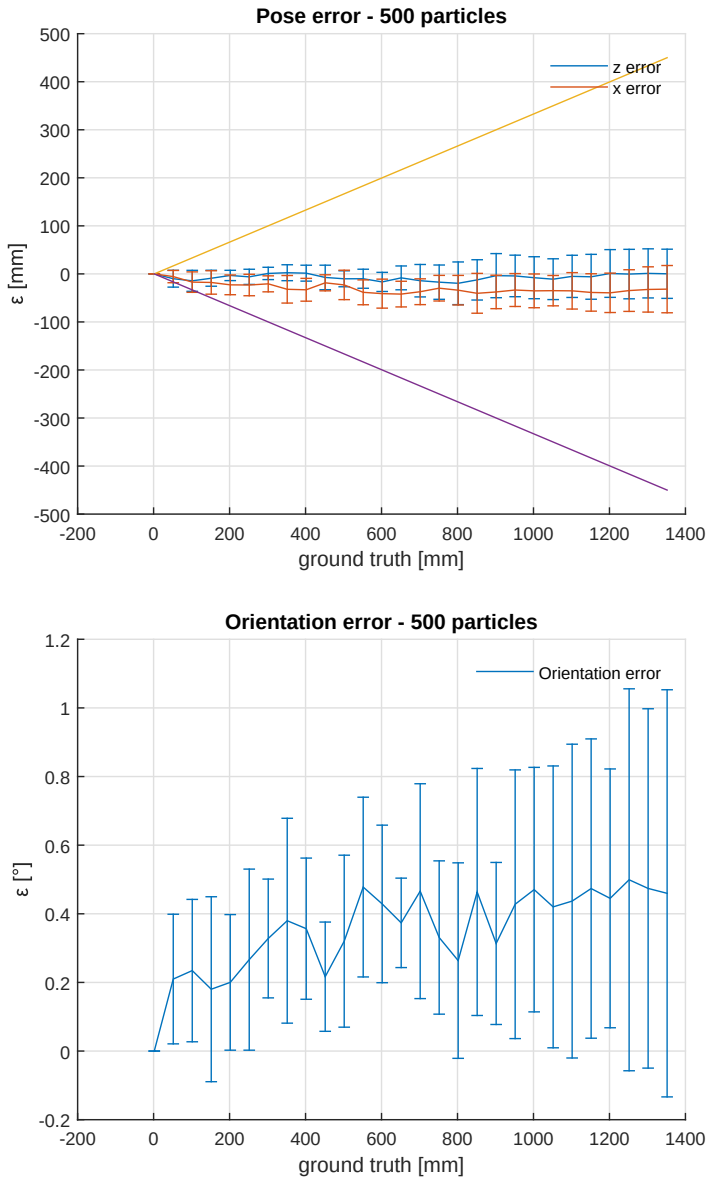


Table 5.6: Particles after the last resampling. Pose and orientation error vs imposed displacement. Oblique lines for odometry error bounds at 1 sigma.

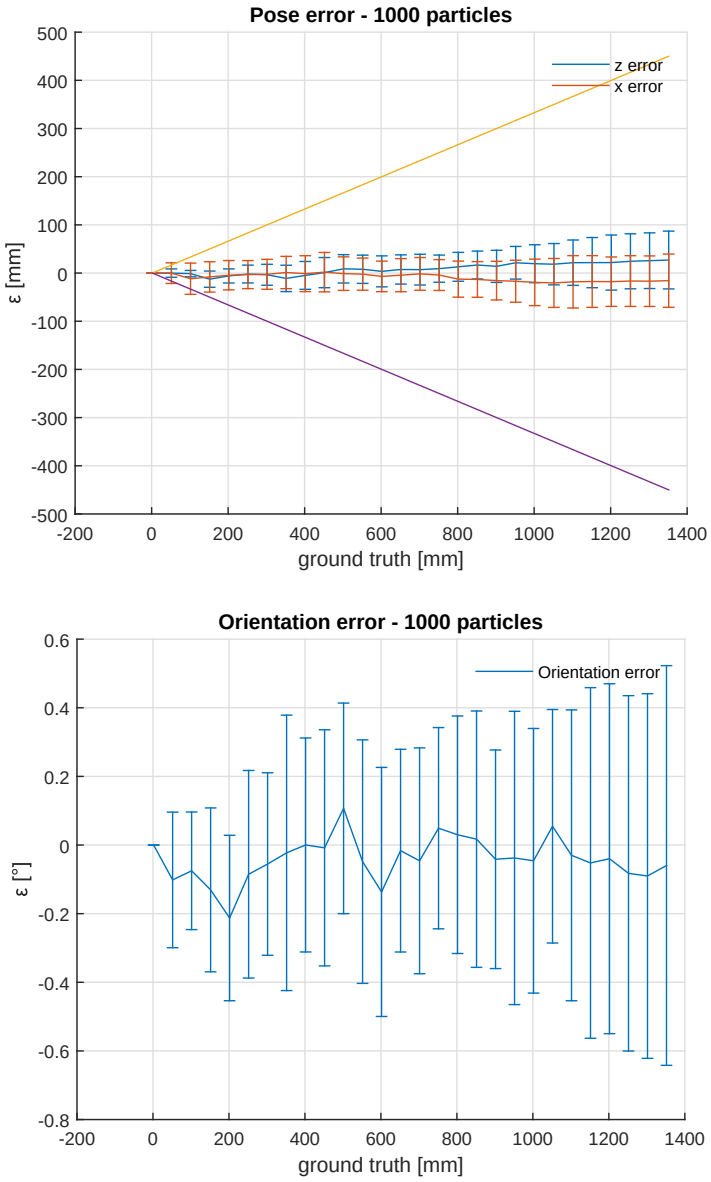


Table 5.7: Particles after the last resampling. Pose and orientation error vs imposed displacement. Oblique lines for odometry error bounds at 1 sigma.

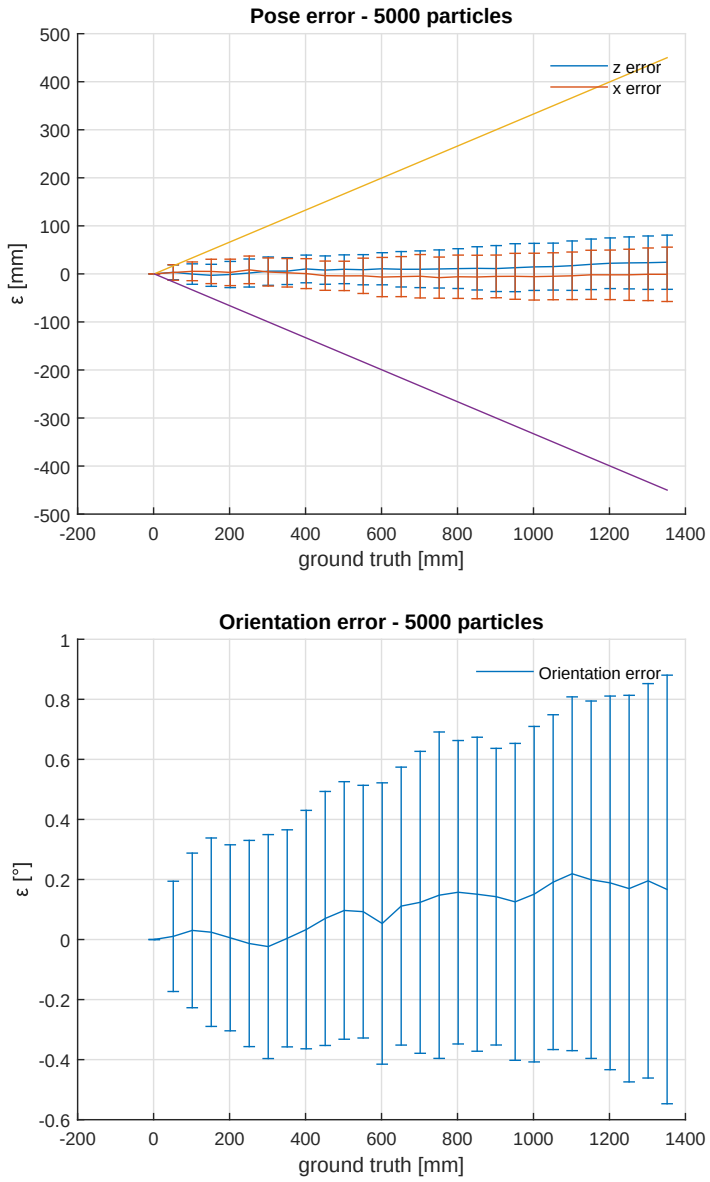


Table 5.8: Particles after the last resampling. Pose and orientation error vs imposed displacement. Oblique lines for odometry error bounds at 1 sigma.

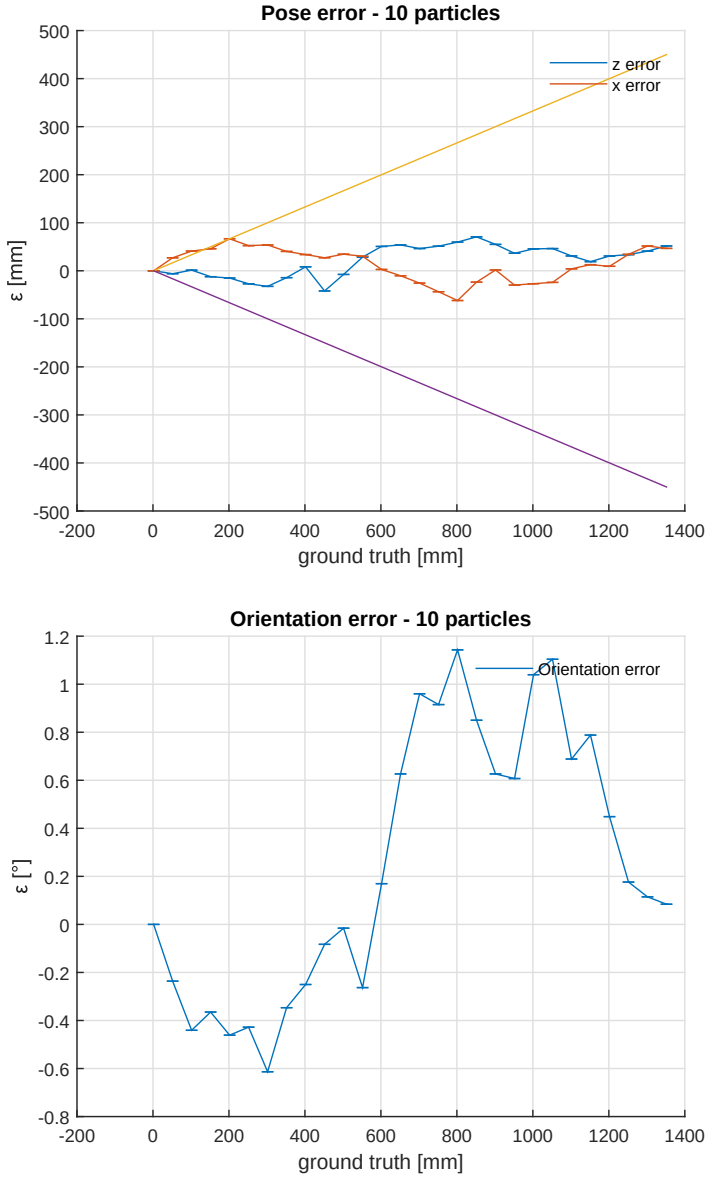


Table 5.9: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.5 (normalized). Oblique lines for odometry error bounds at 1 sigma.

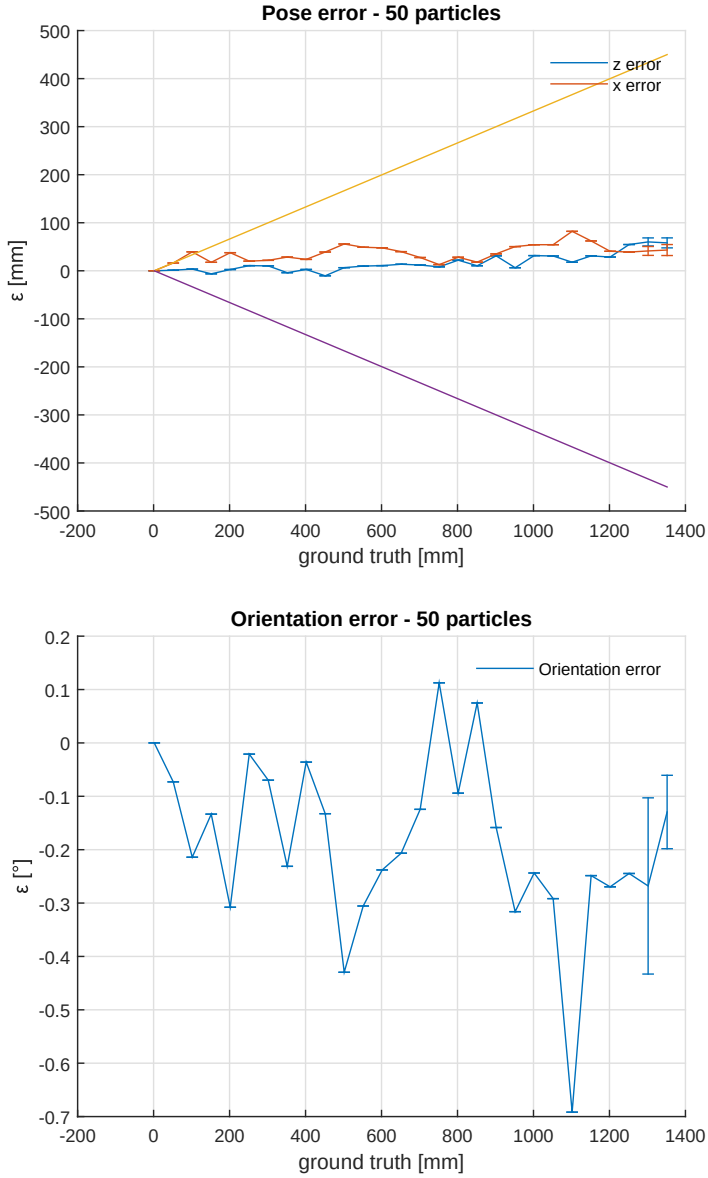


Table 5.10: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.5 (normalized). Oblique lines for odometry error bounds at 1 sigma.

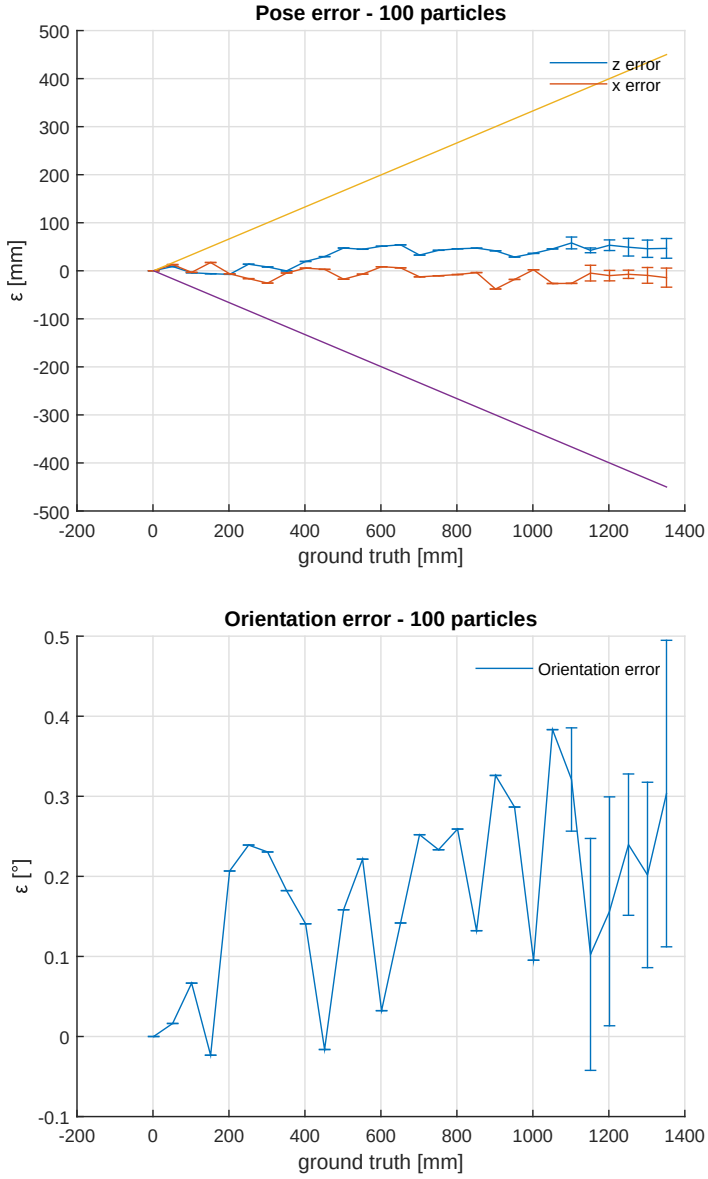


Table 5.11: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.5 (normalized). Oblique lines for odometry error bounds at 1 sigma.

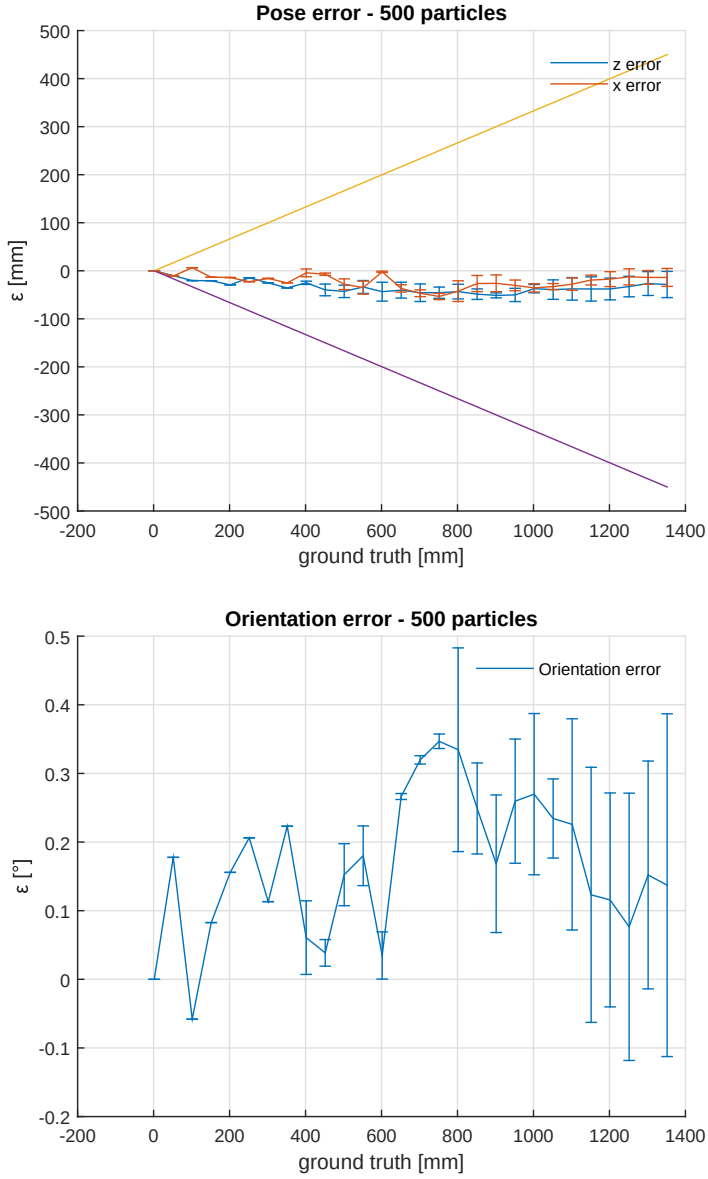


Table 5.12: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.5 (normalized). Oblique lines for odometry error bounds at 1 sigma.

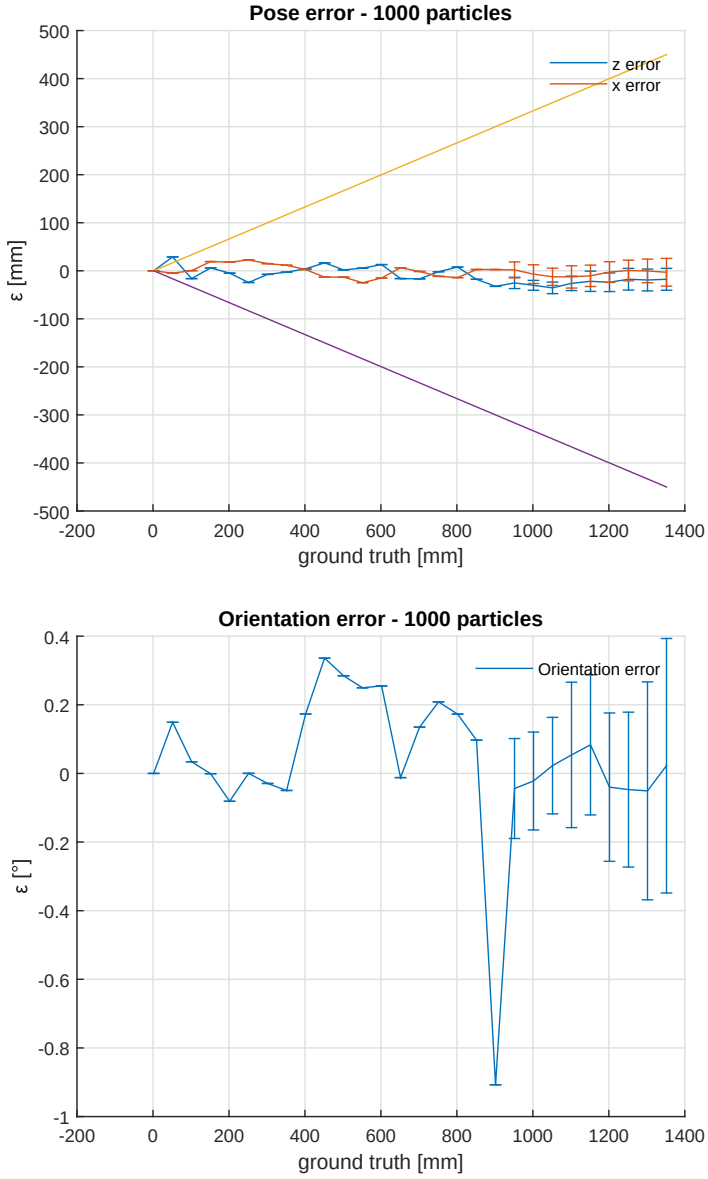


Table 5.13: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.5 (normalized). Oblique lines for odometry error bounds at 1 sigma.

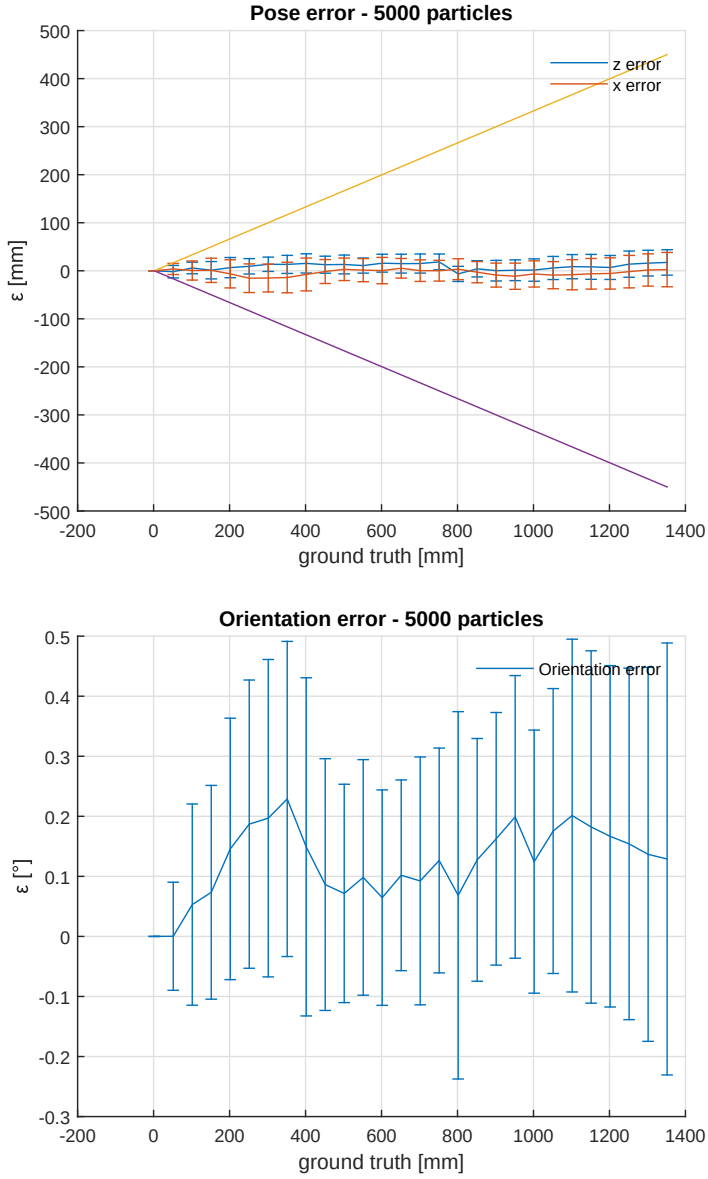


Table 5.14: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.5 (normalized). Oblique lines for odometry error bounds at 1 sigma.

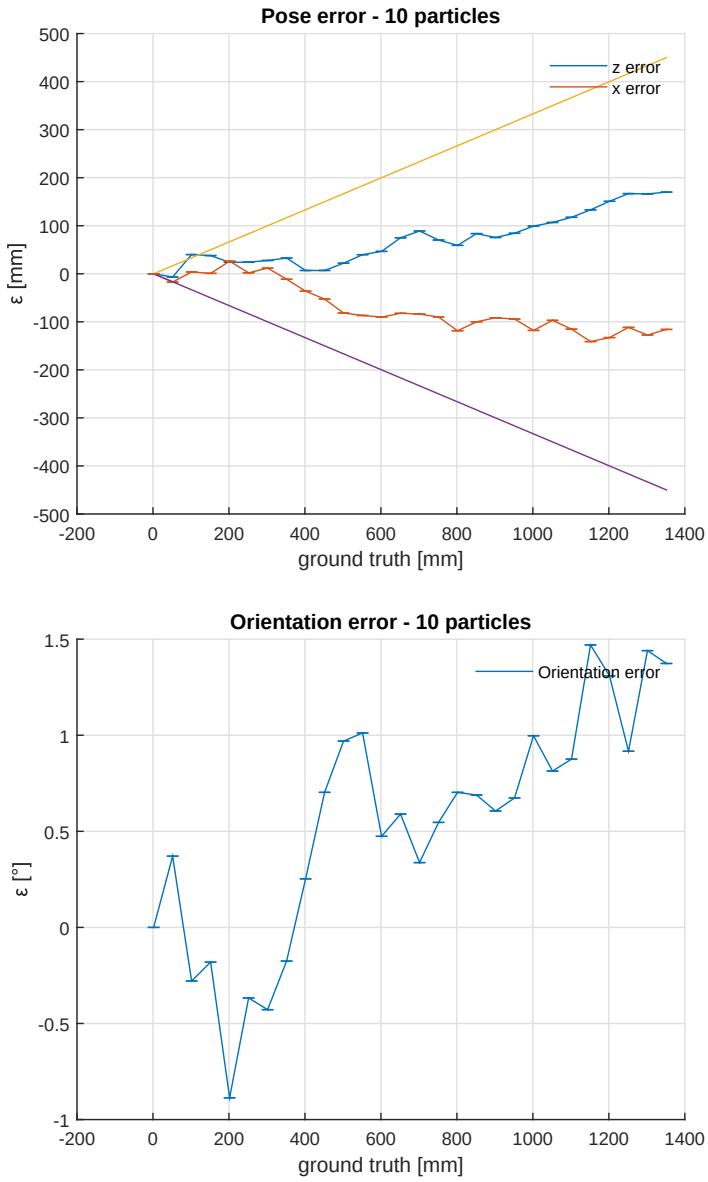


Table 5.15: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.8 (normalized). Oblique lines for odometry error bounds at 1 sigma.

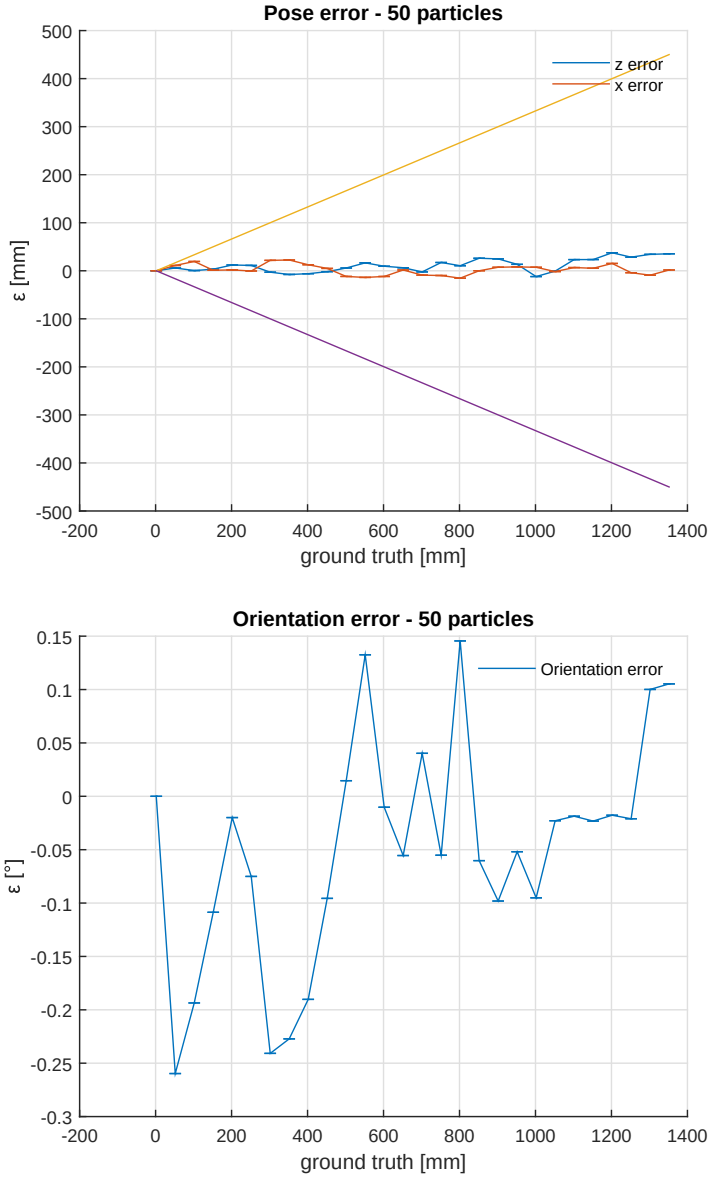
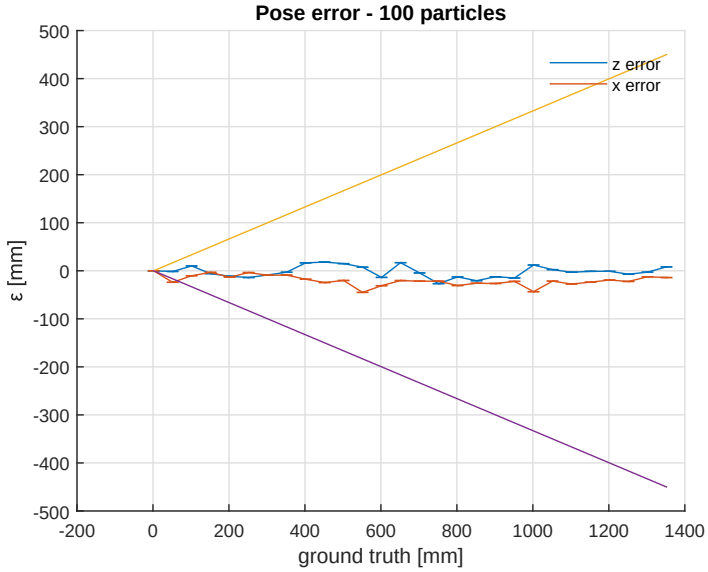


Table 5.16: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.8 (normalized). Oblique lines for odometry error bounds at 1 sigma.



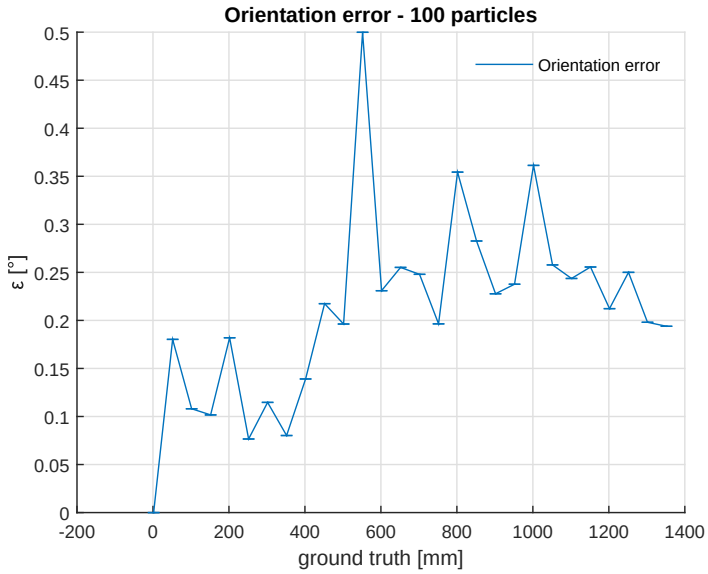


Table 5.17: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.8 (normalized). Oblique lines for odometry error bounds at 1 sigma.

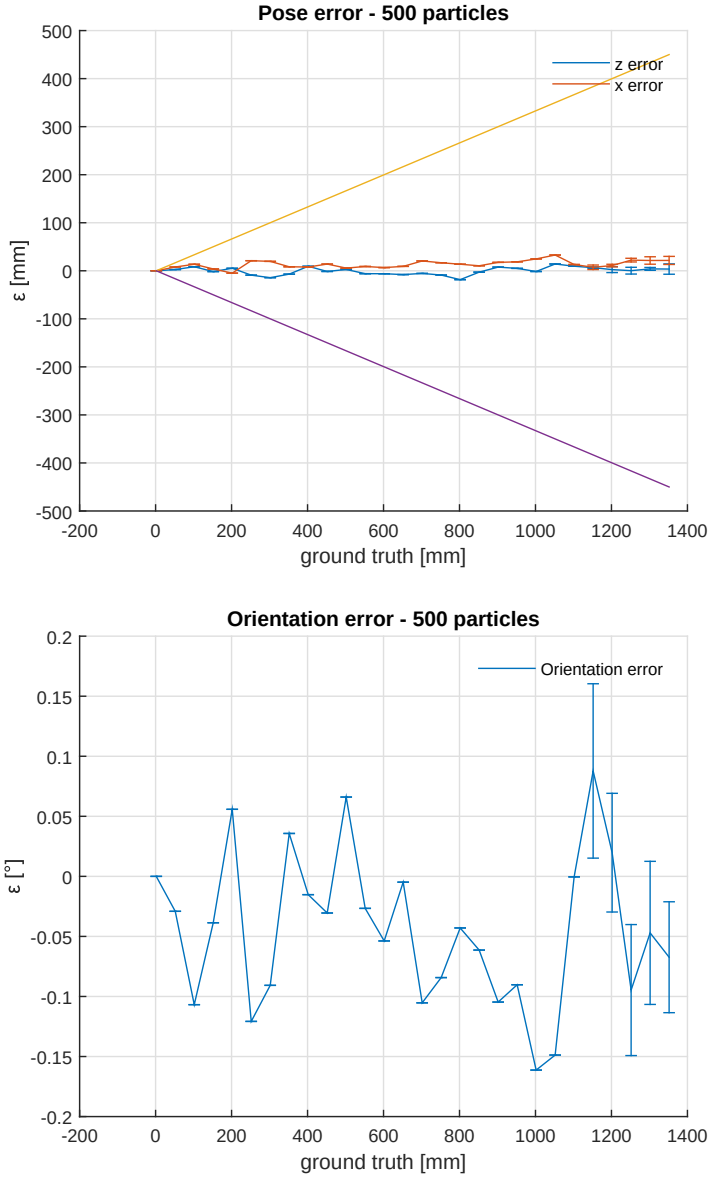


Table 5.18: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.8 (normalized). Oblique lines for odometry error bounds at 1 sigma.

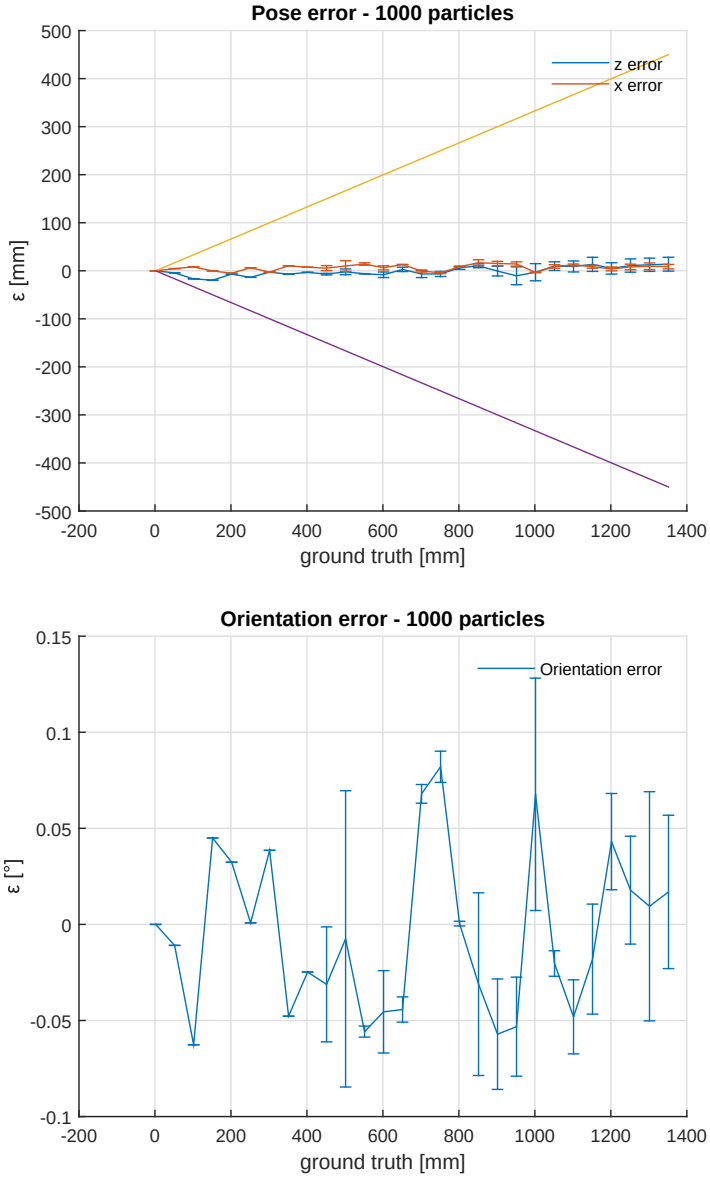


Table 5.19: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.8 (normalized). Oblique lines for odometry error bounds at 1 sigma.

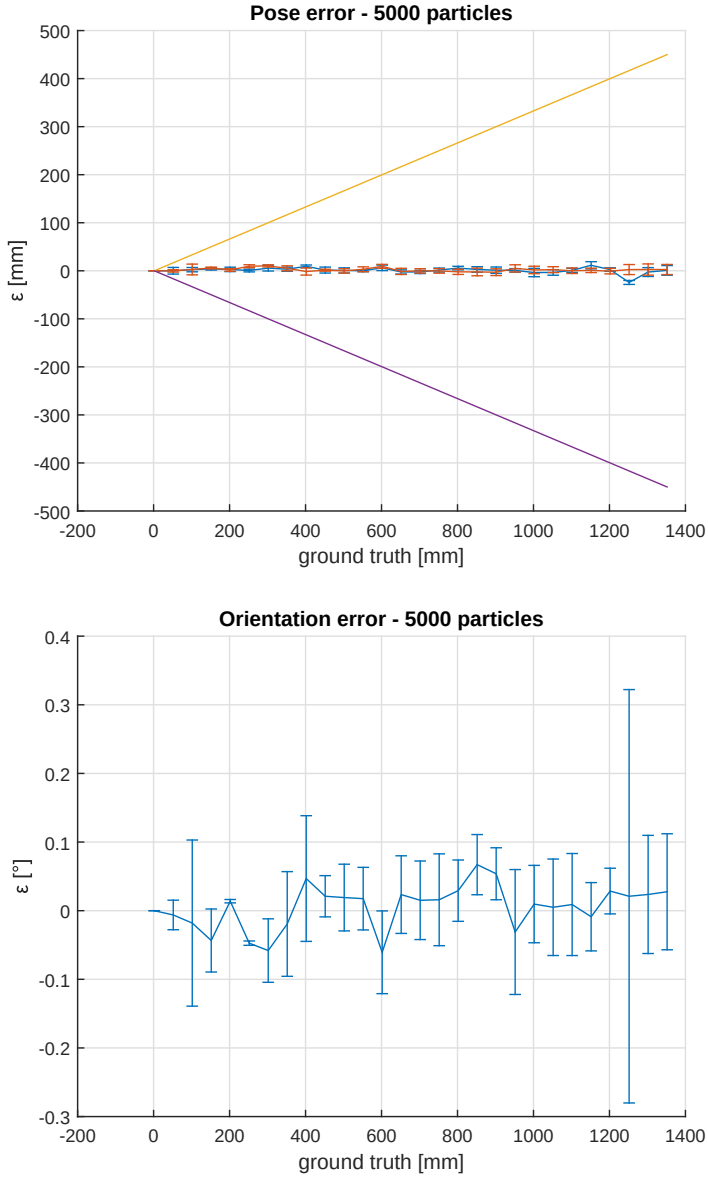


Table 5.20: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.8 (normalized). Oblique lines for odometry error bounds at 1 sigma.

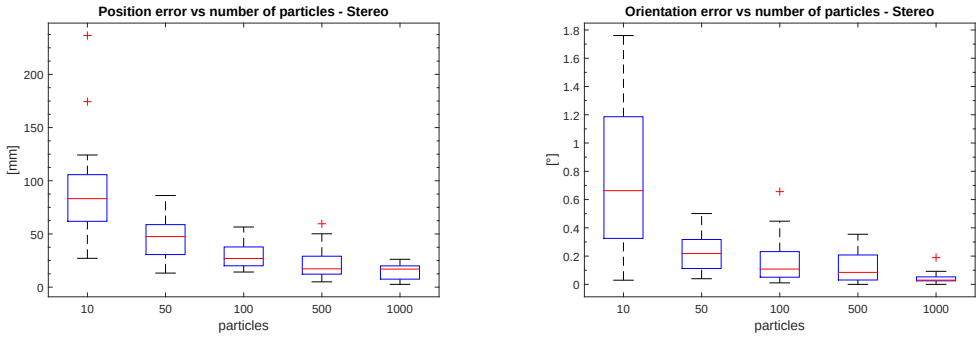


Table 5.21: Averaged results for the mean pose estimated by the stereo algorithm in 20 runs. The red line is the median, the boxes denote the 25-th and 75-th percentiles, the whiskers extends from the lowest to the highest values and the red crosses denote the outliers. For 1000 particles the final position error is lower that 2% of the total imposed displacement and the orientation error is less than 0.1 °

Figure 5.21 shows using a boxplot the averaged final position and orientation errors for 20 runs of the stereo algorithm as a function of the number of particles employed. The results show that the final error decreases as the particle set is more and more dense reaching an asymptote for more than 1000 particles (similarly to [15]). The pose estimation accuracy increases more quickly in relation to the orientation while the increment is less significant for the Euclidean position. That is because changes in viewpoint induce higher innovation errors than changes in position, especially if the camera is moving towards the landmarks.

Mono SLAM - Translational test The monocular SLAM is performed on the images captured by the left camera of the stereo setup. Features are tracked computing the parallax accumulated on every feature to initialize them by triangulation when a minimum parallax of 1° is reached. In literature various methods for delayed and undelayed landmark initialization can be found, since this thesis work is focused on a precise evaluation of pose and orientation estimation ability, robust initialization capabilities of the algorithm are not taken into consideration. Landmark initialization is then performed imposing the correct displacement of the camera for two steps resulting the Euclidean coordinates of a minimum number of landmarks to permit coherent weighting of the particles in the following steps. While in the stereo setup feature triangulation result depth information about the observed landmark, monocular feature estimations carry information only when parallax is accumulated. In this translational motion scenario, features located in front of the camera will not induce significant innovation errors if the motion hypotesis involves displacement in their direction because the projection of the features will not change significantly. The particle weigthing is then very inefficient in this case resulting in a *frontal blindness* that can really degenerate the performance of both motion estimation and map building. On the other hand, viewpoint changes are discriminated very precisely because the landmark projections in the camera sensor travels almost independently to the depth of the points. Lateral movements too induces a more peaked weighted distribution than longitudinal displacements. Being one particle a sample from a multivariate Gaussian distribution, poses very close to the actual location of the camera can be oriented on a wrong angle resulting in a low particle weight. Contrary, poses located far from the ground truth can be correctly oriented. Since the viewpoint orientation is

more discriminative on the particle likelihood than the pose error, after the resampling procedure a high planar sparsification can occur while estimating correctly the camera orientation. For those reasons a high particle density is needed to estimate the path with a good accuracy.

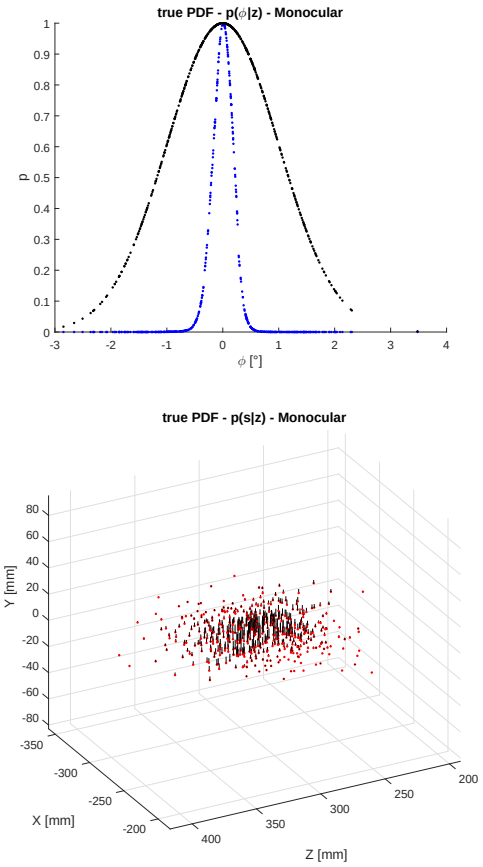


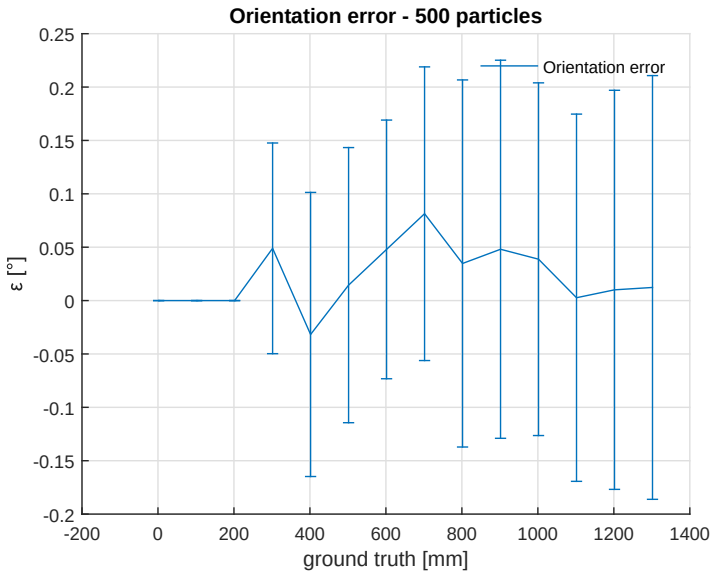
Table 5.22: Proposed distributions for orientation (black) and pose changes (red) treated separately. It is evident on a quality level how the actual distribution of the orientations is much more peaked than the distribution over the camera planar poses. $\sigma_\theta = 1°$, $\sigma_s = 50/3mm$

Looking at the results for the monocular translational test it is evident that the algorithm is less efficient than the stereo algorithm at costraining the trajectory hypotesis over the true path. Because of the low weight variations over the different pose hypotesis, the pose estimation is poorer that the orientation estimation as expected. The error over longitudinal translation is higher that the error of the lateral movement estimation because of a few reasons. First the innovation error of the frontal landmarks is quite independent of longitudinal motion because the projection of the 3D points remain unchanged. Secondly, being the uncertainty of the landmarks much higher in the direction of the preimage line than over the orthogonal directions, the innovation errors for reprojection errors parallel to the highest uncertainty directions are smoothed. Considering the mean values of the planar location and orientation of the particle families that survived the last resampling, the table 5.27 summarizes the error achieved in the best simulation results for this monocular algorithm.

	case 2 ($w_{min} = 0.5$)	case 3 ($w_{min} = 0.8$)
particles	500	1000
ε_s [mm]	11.68 (0.90%)	22.53 (1.29%)
ε_θ [°]	0.012	0.012

Table 5.27: Best results for the monocular algorithm

While a EKF based SLAM is a deterministic computation, particle filter based SLAM algorithms return a stochastic result being the resampling steps a chain of (weighted) casual choiches. The following boxplots show the variation of the mean value for planar error and orientation over 20 runs for the monocular algorithm in relation to the number of particles. It is evident



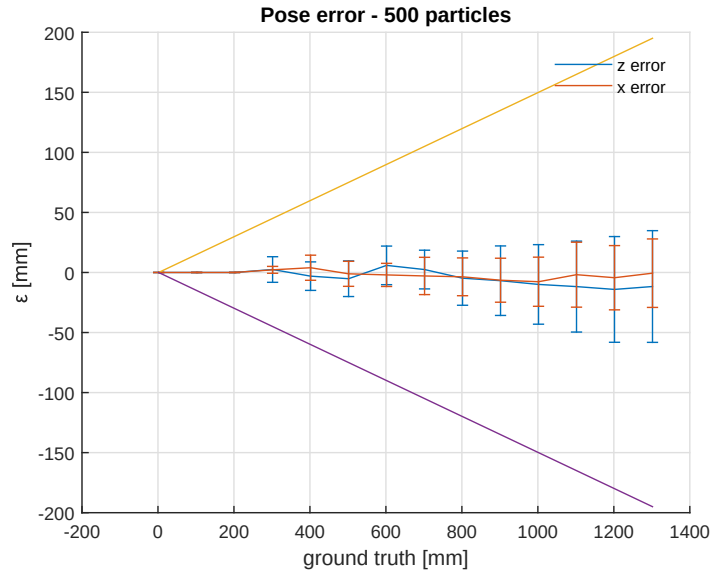


Table 5.23: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.5 (normalized)

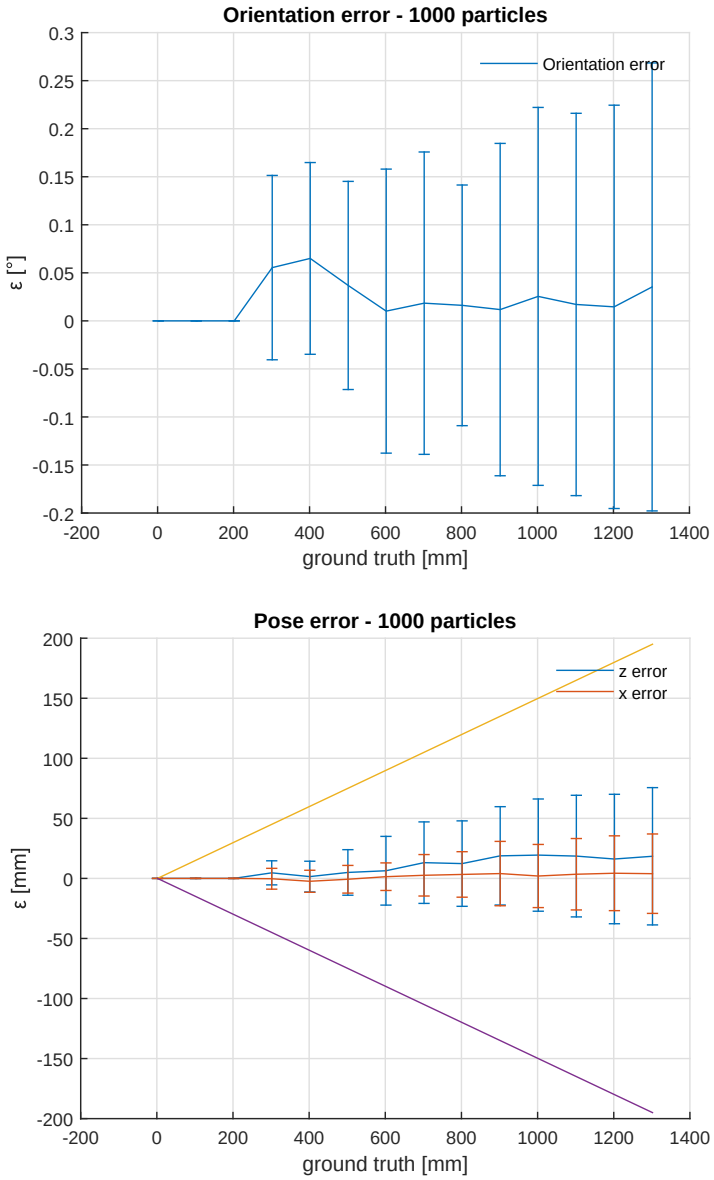
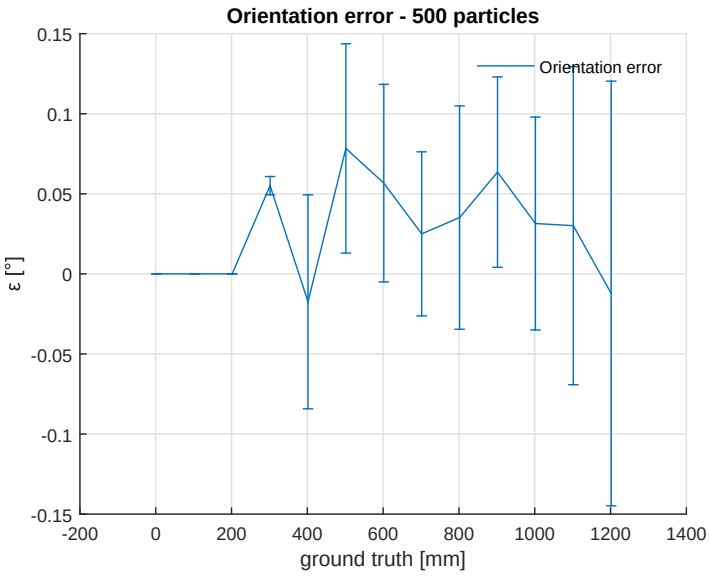


Table 5.24: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.5 (normalized)



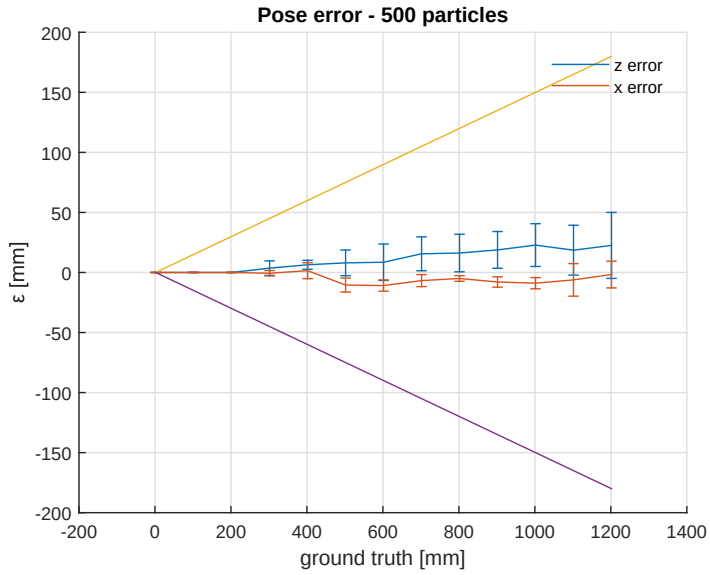
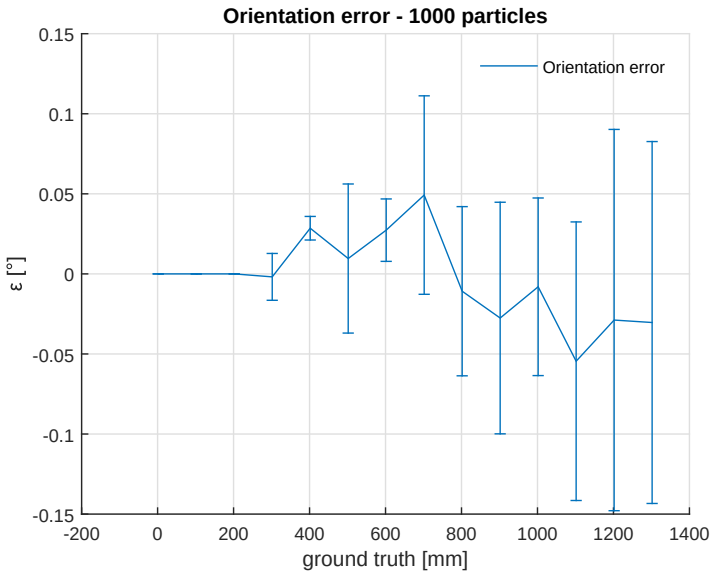


Table 5.25: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.8 (normalized)



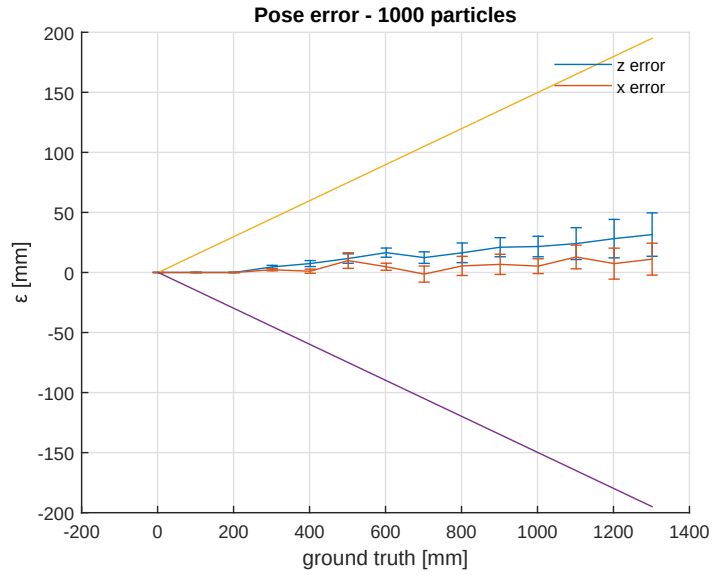


Table 5.26: Particles after the last resampling. Pose and orientation error vs imposed displacement. Resampling with a minimum weigth of 0.8 (normalized)

how the lowest errors can be achieved with a high particle number and how the orientation accuracy improves much faster that the localization accuracy. Figure 5.28 summarizes the number of detected and tracked (via data association) features as well as triangulated landmarks and updated landmarks via EKF.

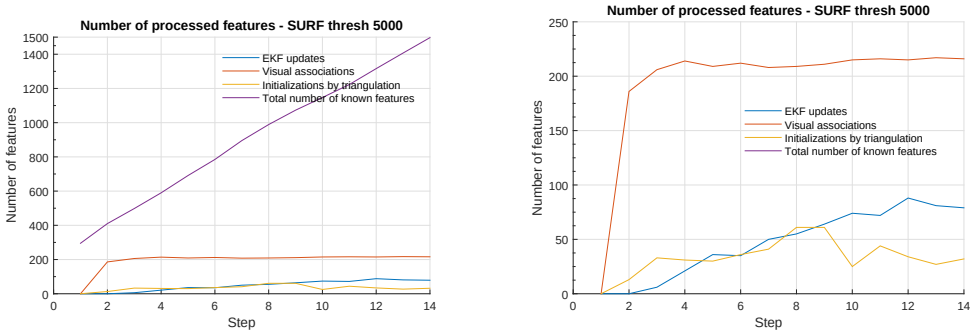


Table 5.28: Number of features involved in the monocular algorithm computation. SURF threshold = 5000.

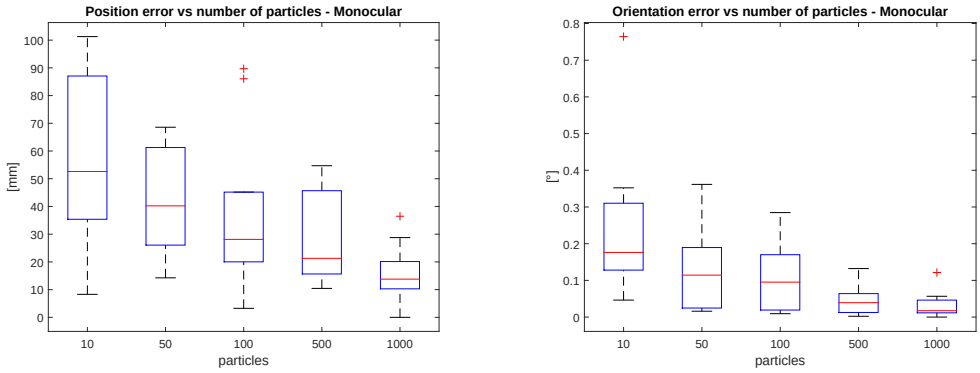


Table 5.29: Boxplots of pose and orientation error vs particle number. The results are obtained averaging 20 runs in the translational test.

Stereo vs Mono map building The results of map building from the monocular and stereo algorithms are now reported and compared. While for stereoscopic landmark evaluations even a single observation can provide reliable informations about the landmark Euclidean location, the monocular algorithm has to correct the first initializations until the uncertainties over the landmark poses are low enough. The locations would otherwise not represent the actual landmark pose in the map giving misleading information about the environment geometry. In the plotted monocular map can be seen the landmarks with the lowest uncertainties. The room in which the image dataset has been captured is approximately 6m wide and 3m tall. The map build by both the algorithms is coherent to the knowledge of the environment geometry. From the top view can be recognized the desks and the central corridor between them. Landmarks are modeled as a Gaussian distribution, the mean is plotted as a red cross and the uncertainty ellipsoid is plotted for 1 sigma in black colour. The camera poses are plotted as blue dots and the viewpoint direction is omitted being non distinguishable from a vector parallel to the Z axis.

Figure 5.2: Glimpse of the viewpoint of the left camera. The blue crosses are detected features and yellow lines are the associated features motion

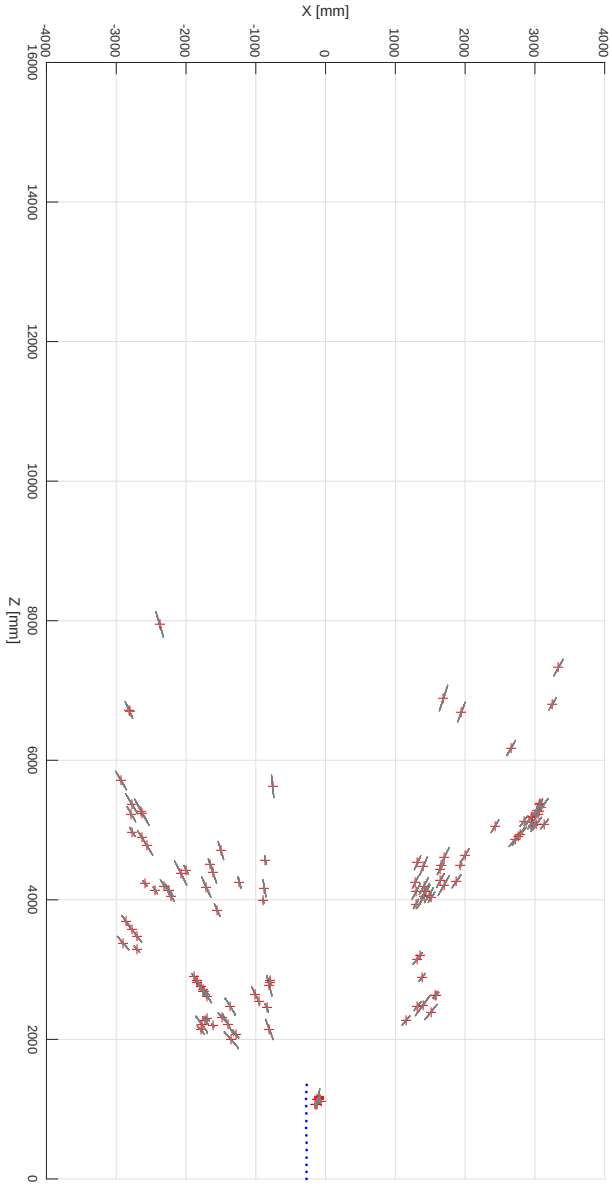


Table 5.30: Top view of the map build by the monocular algorithm over a single translation. The *frontal blindness* of monocular estimations is particularly evident in this image if compared to the stereo map

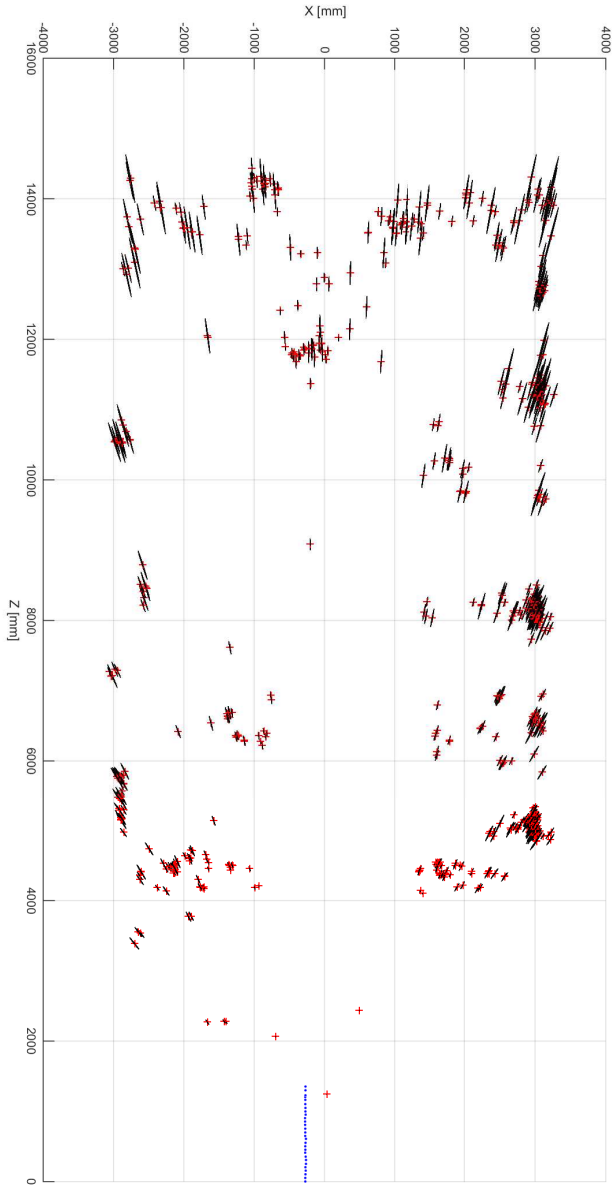


Table 5.31: Top view of the map build by the stereo algorithm over a single translation. This map is more complete than the one built by the monocular algorithm but the first couple of desks is missing.

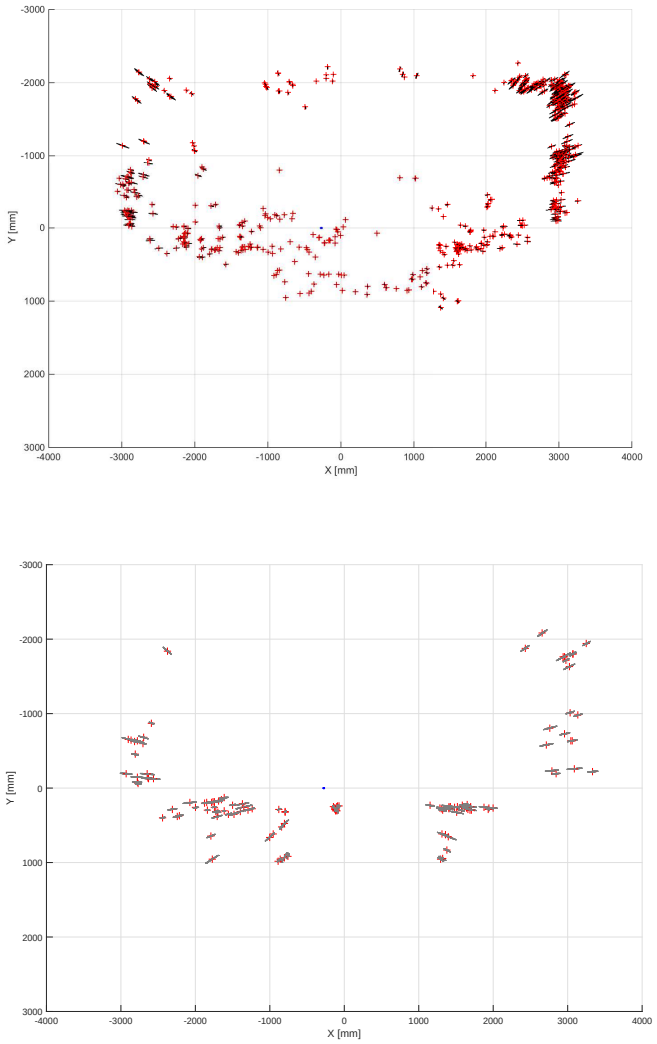


Table 5.32: Frontal view of the map built by the stereo (top) and mono (down) algorithms over a single translation

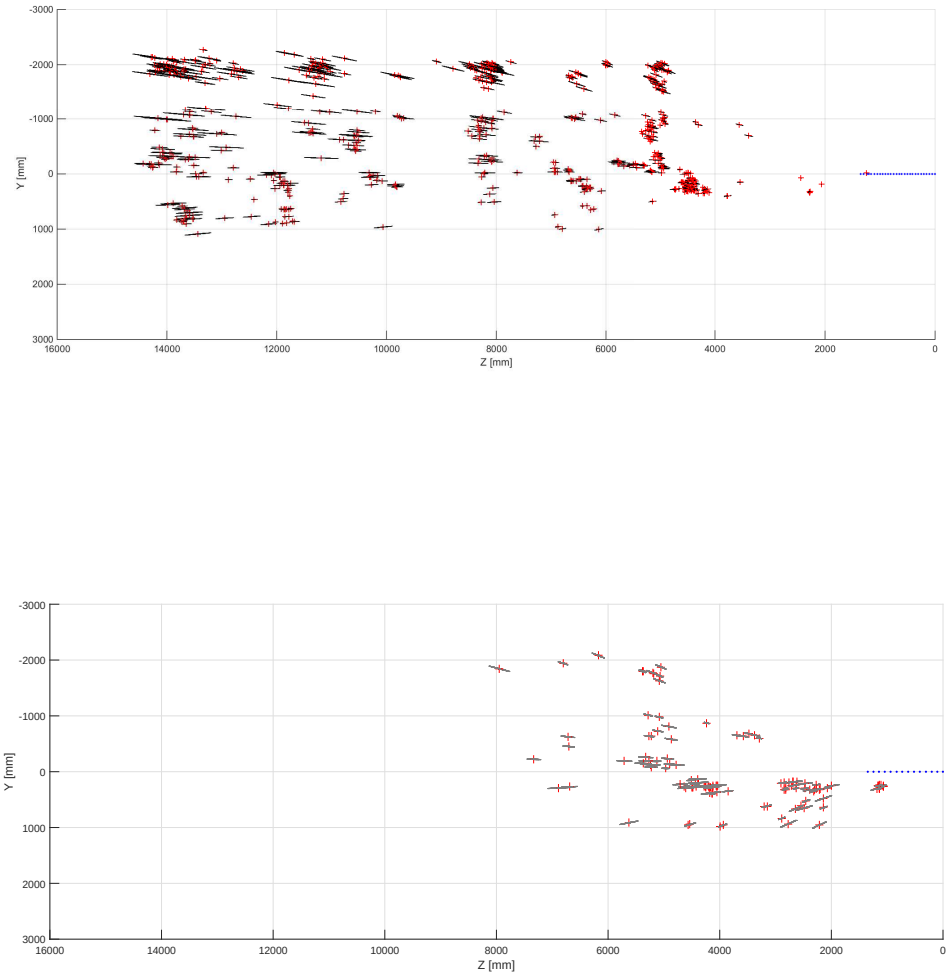


Table 5.33: Side view of the map built by the stereo (top) and mono (down) algorithms over a single translation

Feature detection and matching for Visual Odometry The 3D to 3D visual odometry algorithm explained in section 4.5 returns the rotation matrix and translation vector that describe the motion between two positions in which a set of two images are captured (from left and right cameras). The first task is to detect matching features in all 4 images (left and right for both positions) from which the same points can be triangulated and the relative motion of them estimated via a least squares algorithm. In a visual odometry routine a key aspect is to correctly establish the correlation between feature points via their visual properties, wrong associations leads to wrong motion estimations. Just relying to visual descriptors is not a viable procedure to correlate feature points because similar points can be observed repeatedly in different positions in the image, especially in artificial environments. Matches in left and right images are first obtained by a descriptors distance evaluation (see fig. 5.35) and then refined by a RANSAC procedure to find the best fitting fundamental matrix on the set of correlations, dividing the initial set in a subset of inliers and outliers. This procedure allows to reduce the number of wrong matches to a minimum at the cost of discarding potential right matches. This issue could be minimized imposing a high number of minimum consensus for a model to be considered "good" and reducing it iteratively if the minimum error accepted is not satisfied, this would result in a very high and variable computational effort. In fig. 5.36 the distinction of inliers and outliers operated by the RANSAC algorithm is shown, while almost all the bad matches are rejected, some right matches are also discarded. The final matches set is obtained then finding the ones in the two distinct time steps that are present in both of them therefore referring to the same point in the environment. The final set of matches are depicted in fig. 5.37. Finally some heuristics are applied discarding from the final set the matches where the

n points		before RANSAC		after RANSAC		final
points a	2487	match ab	375	match ab	232	108
points b	2298	match cd	356	match ab	231	
points c	2499	match ac	1639	match ab	1639	
points d	2350					

Table 5.34: Numerical example for a displacement of 50mm, longitudinal translation. Number of SURF points in the 4 images, number of matches after the visual association and after the RANSAC refinement. The final set is the number of correlations used as the input to the visual odometry algorithm after the heuristics have been applied.

distance in pixel from the coordinates of the point in image left position 1 and image left position 2 is very different from the distance of the coordinates of the matching point in image right position 1 and 2. That would be a wrong match in the images at the same time step that survived the epipolar costraint refinement and where one of the points matched in the correlation from image left and position 1 and 2 (green lines in fig. 4.6). The table 5.34 summarize the number of feature points surviving every step of the matching process for a set of images captured at a distance of 50mm. The procedure explained does not guarantee the best performances of the algorithm: generally the more matches are used the better would be the results. Increasing the number of matches by for instance relaxing the thresholds for visual matching and point detection could lead to more false matches which have a destructive influence to the outcome of the computation. Finally, the absence of matching errors does not guarantee good results because they tend to be wrong if little motion of the points is observed.

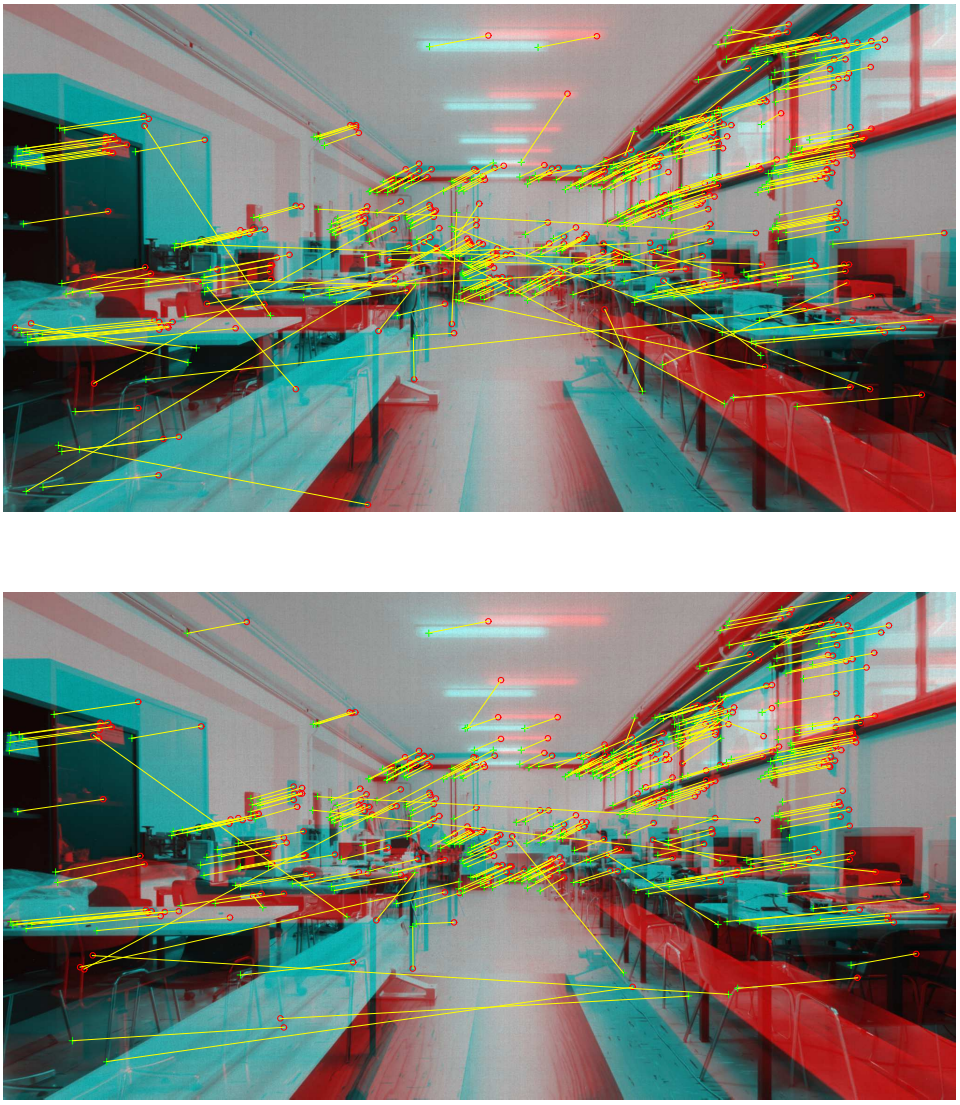
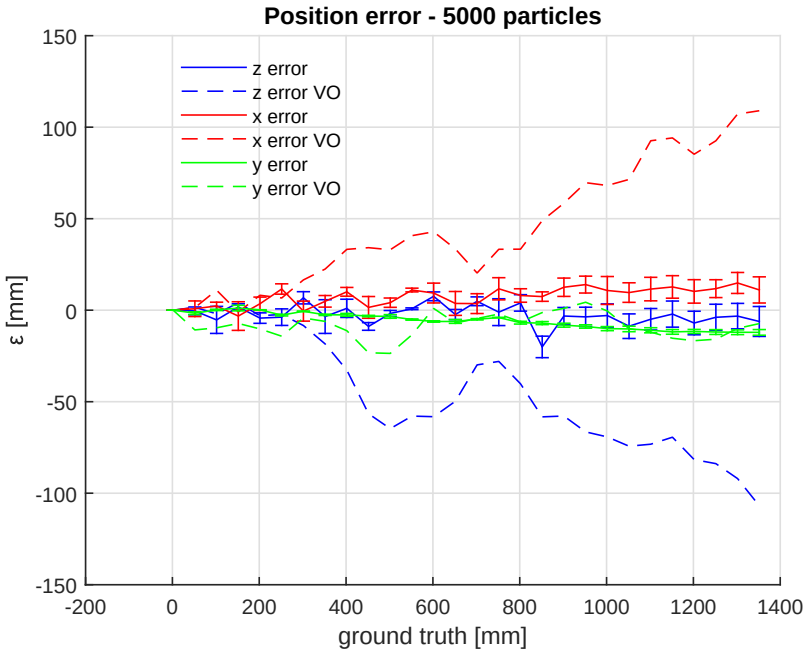


Table 5.35: Superimposed images from left and right cameras and matching features after association of visual descriptors over 50mm imposed displacement. A significant number of wrong association is visible.

Table 5.36: Superimposed images from left and right cameras and matching features after RANSAC refinement of the first set of matches. Almost the totality of wrong matches are discarded but also some correct matches are addressed as outliers. Yellow lines and green crosses are the inliers and black lines and red crosses are outliers.

Table 5.37: Top figure shows matching points from images captured by the left camera over a 50mm displacement, some false visible matches will be discarded by heuristics. The lower figure shows the final set of matching feature points that can be used as an input for the visual odometry algorithm.

Stereo visual SLAM using Visual Odometry input While in the previous analysis the odometry information is simulated imposing a displacement and viewpoint change with a high noise, in this section visual odometry is employed to retrieve motion information for the camera to be corrected by the SLAM algorithm. The SLAM algorithm used is exactly the same as the one used before with the only difference in the motion input function. The visual odometry algorithm is independent of the map relying only on the relative motion of th etriangulated points in two consequent time steps. The odometry information is therefore univocal and viable to be employed indistinctively by every particle. The algorithm is evaluated on a single translational test where the ground reference for every captured frame is known. The visual odometry algorithm is roughly implemented because no RANSAC refinement of the motion parameters on the landmarks reprojection is employed. This is done in order to give noisy inputs to the SLAM algorithm to highlight the ability of choosing good motion hypotesis and therefore retain a low error. The proposed distribution for the camera pose is obtained via a Monte Carlo error propagation through the odometry algorithm. Figure 5.38 show the position and orientation errors accumulated by the visual odometry algorithm alone and the error of the particles that survived the last resampling. A significant reduction of the final pose estimation error is obtained by the SLAM implementation costraining the final error around 1.5%. Figure 5.39 show the map and mean trajectory evaluated by the SLAM algorithm using VO as the only input. While the visual odometry alone does not diverge, the trajectory corrected by the SLAM algorithm appears significantly smoothed and more straight.



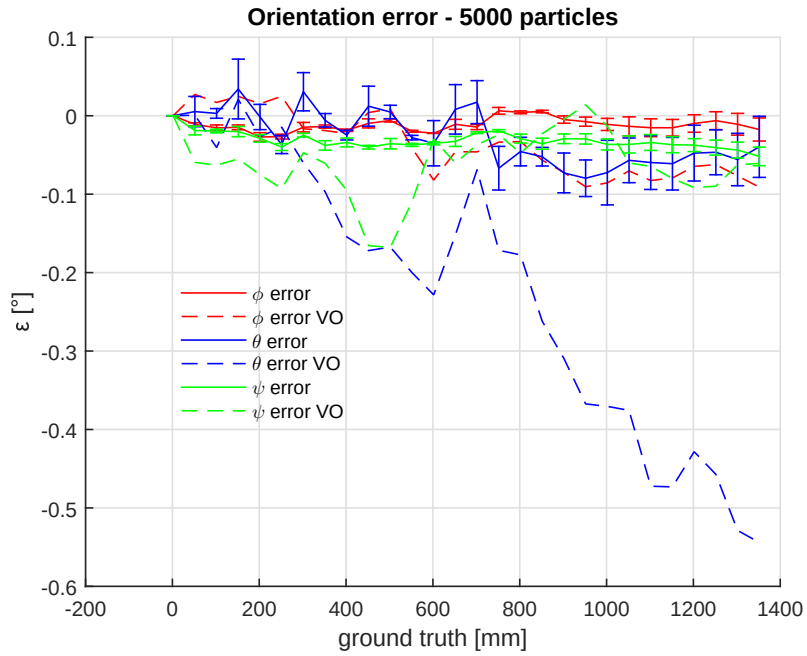


Table 5.38: Pose errors comparison between the visual odometry algorithm alone (dashed lines) and the SLAM algorithm using visual odometry as the only input. Errorbars costrain the particle set around the mean trajectory. Final position error is below 1.5% and the orientation error is below 0.1°. The divergence of the trajectory evaluated by the visual odometry algorithm alone is evident.

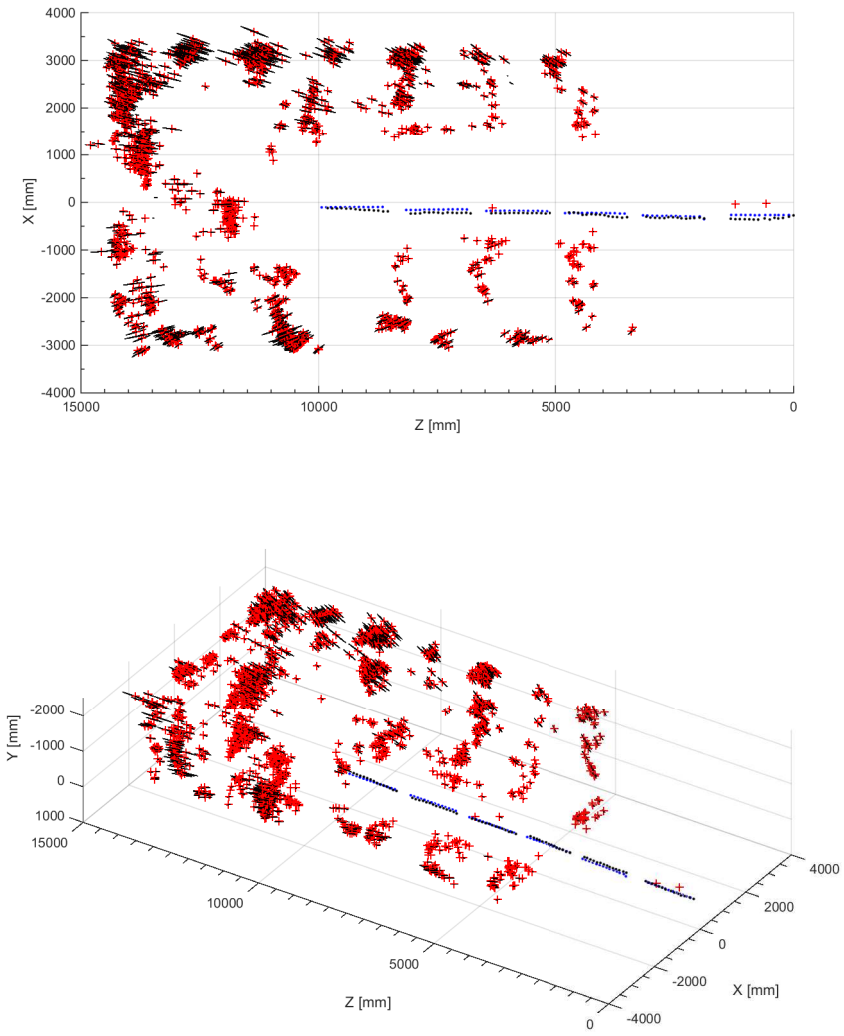


Table 5.39: Top view and perspective view of the full map and trajectory of the camera. Blue dots for the SLAM corrected trajectory and black dots for the trajectory evaluated by the visual odometry algorithm. No ground thruth is available.

Chapter 6

Conclusions

The algorithms implemented and tested in this thesis work for particle filter based stereo and monocular visual SLAM have proven to be effective in reducing the uncertainty associated to odometry information and correctly estimating the path of the cameras. Similar errors at the end of the translational path involved in the analysis have been obtained by both the algorithms but the monocular SLAM required a denser particle sparsification than the stereo SLAM algorithm. The environment observed by the cameras represented a though challenge in path estimation for monocular vision because of the little-to-none parallax changes in the tracking of the image features. From the images of the map built by the monocular algorithm it can be clearly seen the effect of frontal blindness that monocular vision suffers from. The results obtained show that a greater scale implementation is feasible for both wheeled vehicle exploration using a stereo camera or UAV application using a ground facing camera. To deeply investigate the pros and cons of using a monocular downward facing camera rather than a forward facing camera further work is required to highlight variations in trajectory estimation accuracy. Different angles between the direction of motion and

the optical axis should be evaluated, from the parallel direction of the optical axis to the orthogonal direction as well as intermediate directions as 30°and 60°.

The results obtained from the SLAM implementation using Visual Odometry as the source of prior motion estimates also shows that it is feasible to use a stereo camera as the only sensor employed in a robot. By implementing a more refined Visual Odometry algorithm it should be feasible for a robot to correctly estimate the trajectory over a larger scale path and also perform loop closure (which is not addressed in this thesis work). For UAV applications, monocular Visual Odometry should also be employed to build a monocular visual SLAM algorithm using one camera as the only sensor. Various tests should then be performed by implementing the algorithm on a drone platform using a wide angle camera to localize the vehicle and map an indoor GPS-denied environment or an outdoor natural environment.

Chapter 7

Appendix

In this chapter the pseudocodes for the implemented algorithms are incorporated. Algorithm 5 explain the structure of the visual stereo SLAM implementation while algorithms 7 and 6 show the structure of the proposed monocular implementations.

Algorithm 5 Visual StereoSLAM

1: $givecameraparamstereo()$
2: $giveparamuncertanty()$
3: $M \leftarrow nparticle$ ▷ Number of particles

4: **for** $i = 1 : nstep$ **do** ▷ Main loop
5: $[im1, im2] \leftarrow imread(camera1, camera2)$
6: $[f1, f2] \leftarrow SurfDetector(im1, im2)$
7: $[location1, location2] \leftarrow RANSAC(f1, f2)$ ▷ Matching and RANSAC

8: **for** $j = 1 : nfeat_i$ **do** ▷ Visual Data Association
9: $index \leftarrow matchfeatures(f_{avg,j}, f_{tot})$ ▷ Avg descr. cam1,cam2
10: **end for**

11: **for** $j = 1 : nfeat_i$ **do** ▷ Triangulation - measure
12: $\hat{X}_j^i \leftarrow tr(location1_j, location2_j, param)$
13: $\hat{P}_j^i \leftarrow trinc(location1_j, location2_j, param, inc)$
14: **end for**

15: **for** $m = 1 : M$ **do** ▷ Particle loop

16: $[R_1^i, t_{i1}^1]^{\{m\}} \leftarrow motion([R, t]_{mean}, noise)$ ▷ Camera pos (SDR world)
17: *update pose history*

18: **for** $j = 1 : nfeat_i$ **do** ▷ Feature loop

19: **if** $landmark_j$ *new* **then** ▷ First acquisition

20: *update* f_{tot} *with* f_j

21: *update* $X^{[m]}_{struct,tot}$ *with* $\hat{X}^{1,[m]}_j$

22: *update* $P^{[m]}_{struct,tot}$ *with* $\hat{P}^{1,[m]}_j$

23: $w_j \leftarrow w_0$

24: **else** ▷ EKF update

25: *recall last* $X^{1,[m]}_j$ ▷ SDR world

26: *recall last* $P^{1,[m]}_j$

27: $X^{(-)} \leftarrow X^{i,[m]}_{j,last}$ ▷ landmark j; SDR i; particle m

28: $P^{(-)} \leftarrow P^{i,[m]}_{j,last}$

29: $z \leftarrow \hat{X}_j$ ▷ Measure, line 12

30: $S \leftarrow [I] * P^{(-)} * [I] + P_j$ ▷ Inn covariance, line 13

31: $K \leftarrow P^{(-)} * [I] * S^{-1}$

32: $X^{(+)}_j = X^{(-)}_j + K * (z - X^{(-)}_j)$

33: *update* $X^{[m]}_{struct,tot}$ *with* $X^{(+),1,[m]}_j$

34: *update* $P^{[m]}_{struct,tot}$ *with* $P^{(+),1,[m]}_j$

35: $w_j \propto \exp - \frac{((z - X^{(-)}_j) * S^{-}1 * (z - X^{(-)}_j)^T)}{2}$

36: **end if**

37: **end for**

38: $w^{[m]} = \sum_j w^{[m]}_j$

39: **end for**

40: **Resampling** : $particles_{new} \leftarrow datasample(particles_{old}, w^{[m]})$

41: **end for**

Algorithm 6 Visual MonoSLAM. Vision based EKF update

1: $givecameraparam()$
2: $giveparamuncertanty()$
3: $M \leftarrow nparticle$ ▷ Number of particles

4: **for** $i = 1 : nstep$ **do** ▷ Main loop
5: $im \leftarrow imread(camera)$
6: $[f, location] \leftarrow SurfDetector(im)$ ▷ descriptors and pix coord $\{u, v\}$

7: **for** $j = 1 : nfeat_i$ **do** ▷ Visual Data Association
8: $index \leftarrow matchfeatures(f_j, f_{tot})$
9: **end for**

10: **for** $m = 1 : M$ **do** ▷ Particle loop

11: $[R_1^i, t_{i1}^1]^{\{m\}} \leftarrow motion([R, t]_{mean}, noise)$ ▷ Camera pos (SDR world)
12: $update\ trajectory$
13: **for** $j = 1 : nfeat_i$ **do** ▷ Landmark loop
14: **if** $landmark_j\ new$ **then** ▷ First acquisition
15: $\lambda \leftarrow \lambda_0$ ▷ Default depth
16: $X_j^{i,\{m\}} \leftarrow projectInv(location, lambda, param)$
17: $P_j^{i,\{m\}} \leftarrow projectCov(location, lambda, param, inc)$
18: $update\ X_{struct,tot}^{\{m\}}\ with\ X_j^{1,\{m\}}$
19: $update\ P_{struct,tot}^{\{m\}}\ with\ P_j^{1,\{m\}}$
20: $update\ f_{tot}\ with\ f_j$
21: $w_j \leftarrow w_0$

22: **else** ▷ EKF update

23: *recall last* $X_j^{1,\{m\}}$ ▷ SDR world

24: *recall last* $P_j^{1,\{m\}}$

25: $X^{(-)} \leftarrow X_{j,last}^{i,\{m\}}$ ▷ landmark j; SDR i; particle m

26: $P^{(-)} \leftarrow P_{j,last}^{i,\{m\}}$

27: $z \leftarrow location_j$ ▷ Measurement (u_j, v_j)

28: $expected \leftarrow project(X^{(-)}, param)$ ▷ Projection on image plane

29: $J \leftarrow projectJac(X^{(-)}, param)$ ▷ Jacobian of projection function

30: $S \leftarrow J * P^{(-)} * J + P_j$

31: $K \leftarrow P^{(-)} * J * S^{-1}$

32: $X^{(+)} = X^{(-)} + K * (z - expected)$

33: $P^{(+)} = ([I] - KJ^T)P^{(-)}$

34: *update* $X_{struct,tot}^{\{m\}}$ *with* $X_j^{(+),1,\{m\}}$

35: *update* $P_{struct,tot}^{\{m\}}$ *with* $P_j^{(+),1,\{m\}}$

36: $w_j \propto \exp -\frac{((z-X_j^{(-)})*S^{-1}*(z-X_j^{(-)})^T)}{2}$

37: **end if**

38: **end for**

39: $w^{\{m\}} = \sum_j w_j$

40: **end for**

41: **Resampling** : $particles_{new} \leftarrow datasample(particles_{old}, w^{\{m\}})$

42: **end for**

Algorithm 7 Visual MonoSLAM. Triangulation based EKF update

1: $givecameraparam()$
2: $giveparamuncertanty()$
3: $M \leftarrow nparticle$ ▷ Number of particles

4: **for** $i = 1 : nstep$ **do** ▷ Main loop
5: $im \leftarrow imread(camera)$
6: $[f, location] \leftarrow SurfDetector(im)$ ▷ Descriptors and pixel coordinatex $\{u, v\}$

7: **for** $j = 1 : nfeat_i$ **do** ▷ Visual Data Association
8: $index \leftarrow matchfeatures(f_j, f_{tot})$
9: **end for**

10: **for** $m = 1 : M$ **do** ▷ Particle loop

11: $[R_1^i, t_{i1}^1]^{\{m\}} \leftarrow motion([R, t]_{mean}, noise)$ ▷ Camera pos (SDR world)
12: $update\ trajectory$
13: **for** $j = 1 : nfeat_i$ **do** ▷ Landmark loop
14: **if** $landmark_j\ new$ **then** ▷ First acquisition
15: $update\ f_{tot}\ with\ f_j$
16: $w_j \leftarrow w_0$

17: **else** ▷ Known feature
18: $\alpha \leftarrow parallax(location(i0)_j, location(i)_j, R_{i_0}^i, t_{i_0,i})$

19: **if** $\alpha > thresh$ **then**

20: **if** *first time triangulated* **then**

21: $X_j^{i,\{m\}} \leftarrow tr(location(i0)_j, location(i)_j, param)$ ▷ Triang

22: $P_j^{i,\{m\}} \leftarrow trinc(location(i0)_j, location(i)_j, param)$ ▷ State covariance

23: *update* $X_{struct,tot}^{\{m\}}$ *with* $X_j^{1,\{m\}}$

24: *update* $P_{struct,tot}^{\{m\}}$ *with* $P_j^{1,\{m\}}$

25: **else**

26: *recall last* $X_j^{1,\{m\}}$ ▷ SDR world

27: *recall last* $P_j^{1,\{m\}}$

28: $X^{(-)} \leftarrow X_{j,last}^{i,\{m\}}$ ▷ landmark j; SDR i; particle m

29: $P^{(-)} \leftarrow P_{j,last}^{i,\{m\}}$

30: $z \leftarrow location_j$ ▷ Measurement (u_j, v_j)

31: $expected \leftarrow project(X^{(-)}, param)$ ▷ Projection on image plane

32: $J \leftarrow projectJac(X^{(-)}, param)$ ▷ Jacobian of projection function

33: $S \leftarrow J * P^{(-)} * J + P_j$

34: $K \leftarrow P^{(-)} * J * S^{-1}$

35: $X^{(+)} = X^{(-)} + K * (z - expected)$

36: $P^{(+)} = ([I] - KJ^T)P^{(-)}$

37: *update* $X_{struct,tot}^{\{m\}}$ *with* $X_j^{(+),1,\{m\}}$

38: *update* $P_{struct,tot}^{\{m\}}$ *with* $P_j^{(+),1,\{m\}}$

$w_j \propto \exp -\frac{((z-X_j^{(-)})*S^{-}1*(z-X_j^{(-)})^T)}{2}$

$i0_j \leftarrow i$

end if

end if

end if

end for

$w^{\{m\}} = \sum_j w_j$

end for

Resampling : $particles_{new} \leftarrow datasample(particles_{old}, w^{\{m\}})$

end for

Book References

[1] Yi Ma, Stefano Soatto, Jana Kosecka, S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Interdisciplinary Applied Mathematics, Volume 26. Springer, 2010.

Article References

[2] Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. 1980.

[3] J. Engel, J. Stuckler, D. Cremers. Large-scale direct SLAM with stereo cameras.

[4] G. Dissanayake, P. Newmann, H. F. Durrant-Whyte, S. Clark, M. Csobra. A solution to the simultaneous localisation and mapping (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 4(3):229–241.

[5] G. Kootstra, S. de Jong, D. Wedema. Comparing the EKF and fast-SLAM solutions to the problem monocular simultaneous localization and mapping. 7 2009.

[6] N. M. Kwok, Q. Ha, G. Fang. Data association in bearing-only SLAM using a cost function-based approach. *Proceedings on IEEE International Conference on Robotics and Automation*, 5 2007.

[7] C. Harris and M. Stephens. A combined corner and edge detector. *The British Machine Vision Conference (BMVC)*, 1988.

[8] J. Sola, A. Monin, M. Devy, T. Lemaire. Undelayed initialization in bearing only SLAM.

[9] P. Elinas, J. J. Little. Stereo vision SLAM: Near real-time learning of 3d point-landmark and 2d occupancy-grid maps using particle filters.

[10] S. Se, D. Lowe, J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The International Journal of Robotics Research*, 21.

[11] M. Li, B. Hong, Z. Cai, R. Luo. Novel rao-blackwellized particle filter for mobile robot slam using monocular vision. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 2.

[12] C. Olson, L. H. Matthies, M. Schoppers, M. W. Maimone. Rover navigation using stereo ego motion. 43:215–229.

[13] Y. Cheng, M. W. Maimone, L. Matthies. Visual odometry on the mars exploration rover. *IEEE Robotics and Automation Magazine*, 6 2016.

[14] N. Y. Khan, B. McCane, and G. Wyvill. Sift and surf performance evaluation against various image deformations on benchmark dataset. *2011 International Conference on Digital Image Computing: Techniques and Applications*, 2011.

[15] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[16] S. Thrun, M. Montemerlo. The graphslam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6).

[17] J. Civera, A. J. Davison, J. M. Martinez Montiel. Inverse depth parametrization for monocular slam. *IEEE Transactions on Robotics*, 24.

[18] K. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 1999.

[19] D. Davison, D. W. Murray. Simultaneous localisation and map-building using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):865–880.

[20] P. H. Torr, D. W. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 24.

[21] Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. *Improving Data Association in Vision-based SLAM*, 7 2006.

[22] C. Gamallo, M. Mucientes, C. V. Regueiro. Visual fastSLAM through omnivision. *Taros*, 2009.

[23] A. J. Davison, I. D. Reid, N. D. Molton, O. Stasse. MonoSLAM: real-time single camera SLAM. *IEEE Transaction of Pattern Analysis and Machine Intelligente, Vol.29 No.6*, 6 2007.

[24] C. Brand, M. J. Schuster, H. Hirschmuller, M. Suppa. Submap matching for stereo-vision based indoor/outdoor SLAM. *IEEE International Conference on Intelligent Robots and Systems*, 2015.

[25] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *9th European Conference on Computer Vision*, 2006.