Embodied Localisation and Mapping

Jakob Schwendner

Fachbereich 3 (Mathematik & Informatik)

Universität Bremen

Dissertation zur Erlangung des Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

Oktober 2013

Gutachter:
Prof. Dr. Frank Kirchner
Prof. Dr. Udo Frese

Datum des Kolloquiums: 20. Dezember 2013

Weitere Mitglieder des Prüfungsausschusses:
Prof. Dr. Joachim Hertzberg
Prof. Dr. Rolf Drechsler
Dr. Yohannes Kassahun
Sascha Arnold

Acknowledgements

First and foremost, I would like to thank my supervisor Frank Kirchner for providing the guidance, support and boundary conditions to make this work possible. The help and feedback was well beyond what could normally be expected, and I am grateful especially for the directioning towards less treaded paths.

This thesis is the result of many years of work at the Robotics Innovation Center (RIC) at the German Research Center for Artificial Intelligence (DFKI). The stimulating work environment at the lab can be attributed to the general tendency of co-workers to be supportive and inspirational. In Sylvain Joyeux I have found a dear guide with exceptional technical and intellectual skills, who has provided me with enduring support and good advice throughout all phases of the generation of the dissertation.

The work presented here has in large parts been performed in the context of the "Intelligent Mobility" Project, which was funded by the Federal Ministry for Economics and Technology (BMWI) through the German Space Agency (DLR) grant number 50 RA 0907. I would like to acknowledge the good work Sylvain Joyeux, Janosch Machowinski, Felix Grimminger, Patrick Paranhos, Christopher Gaudig, Ajish Babu and the student members did for the iMoby project.

The experiments with the SpaceClimber system were performed with the kind support from Sebastian Bartsch.

Further I would like to thank Udo Frese for inspirational discussions, good advice especially on the matter of SLAM as well as agreeing to be my second evaluator.

Jan Albiez and Yohannes Kasahun provided me with moral support and general advice on how to survive the thesis writing.

Lastly mentioned, and most important, is my Family.

Abstract

Mobile autonomous robots have finally emerged from the confined spaces of structured and controlled indoor environments. To fulfil the promises of ubiquitous robotics in complex outdoor environments safe navigation is a key requirement. The research in the simultaneous localisation and mapping (SLAM) community has largely focused on using optical sensors to solve this problem. The fact that the robot is a physical entity which interacts with the environment has largely been ignored. In this thesis a method for localisation and mapping is proposed, which takes the embodiedness of robots into account. A Rao-Blackwellized Particle Filter is used with embodied measurement and prediction models to generate 3D map segments of outdoor environments. Using a hierarchical approach, these segments are combined in a pose graph with explicit global constraints. The errors of the optimized pose graph are back-propagated onto the local map segments, resulting in maps with smooth transitions along the path of the robot. The proposed method is experimentally verified in simulation and on two different physical outdoor systems. The results show that the approach is viable and that the rich modeling of the robot interaction with its environment provides a new modality for improving existing solutions and extending to scenarios where visual processing is not feasible.

Contents

Nomenclature

Roman Symbols

$\mathcal{C}$ Set of points where the body is in contact with the environment, expressed in robot frame B

$\mathcal{N}(\mu, \Sigma)$ Normal distribution with mean μ and covariance Σ

B Robot body fixed frame of reference. The axis are aligned such that x to the right, y forward and z is up.

$C(x_k)$ Full transform $C_{B_k}^{W}$ as given by the state x_k

$C_{B_k}^{A_{k+1}}$ Full transform from frame B at time k to frame A at time $k+1$

e_x, e_y, e_z Unit vector along x, y and z respectively

$R_z^{-1}(C)$ rotation angle around the z axis of the transform C

$R_{B_k}^{A_{k+1}}$ Rotation from frame B at time k to frame A at time $k+1$

$R_z(\theta)$ Rotation transform that rotates by the angle θ around the z-axis

$T_{B_k}^{A_{k+1}}$ Translation from frame B at time k to frame A at time $k+1$

W Geo-fixed reference frame, where y is north, x is east and z is up.

x_k System state, defined as $(x, y, z, \phi, \psi, \theta)$ includes translation and rotation.

m(p,l) map function, which provides a surface patch $s \in S$ for a point p and a search interval l

Chapter 1

Introduction

1.1 Background

For centuries visionary people have dreamt up machines that are able to perform complex tasks for them.

> "If every tool, when ordered, or even of its own accord, could do the work that befits it ... then there would be no need either of apprentices for the master workers or of slaves for the lords." (Aristotle, 322 B.C.)

Even with the limited technical abilities available at the time, first designs for autonomous machines like the Pigeon by Archytas of Tarentum, were envisaged as early as 350 B.C. Great engineers like Leonardo da Vinci (Rosheim [2006]) built mechanical systems that where designed to exhibit complex behaviour not previously attributed to inanimate objects. The idea of self-operating machines has also been employed by a number of writers. The term *robot* originally stems from a 1920 play by Czech writer Karel Čapek, and was later popularized by the science fiction writer Isaac Asimov. The possibility of machines with human-like intelligence seemed to be within reach in what was later called the golden years of Artificial Intelligence (AI) research between 1956 and 1974 (Russell and Norvig [2010]). The employed top-down approach and the focus on expert systems and symbolic reasoning did however not meet its expectations, and finally lead to funding cuts and a forced rethinking of the field. One alternative route was

proposed in the late 80s at MIT, mainly driven by people like Rodney Brooks or Valentino Braitenberg. The underlying idea is that symbolic reasoning is meaningless without a connection to the real world. Brooks argued that artificial intelligence is better defined by the intelligent behaviour of agents that act in a real world. He employed a bottom-up approach in his Behaviour-Based robotics and found that an agent can solve relatively complex tasks using a combination of simple behaviours.

> "The world is its own best model. It is always exactly up to date. It always has every detail there is to be known. The trick is to sense it appropriately and often enough." (Rodney Brooks, 1990)

This view has a close connection to the field of embodied cognition. The premise of this multidisciplinary field is the notion that the body gives rise to the mind, and that all aspects of the mind are governed by the body. Interesting here is the close connection to philosophy, and the body-mind problem. The experiments from Brooks and others have provided new arguments on the philosophical side (Clark [1998]).

The prevalent view of researcher in the field of AI today is that a combination of both bottom-up and top-down methods are required to get closer to the vision of rational agents that exhibit intelligent behaviour in complex environments (Russell and Norvig [2010]). The advances in the mechanical engineering fields have provided systems that are very capable in static environments. Aristotle's tools that could do the work on their own and would make slaves unnecessary have become a reality in a number of areas. The degree of automation in industrial environments today is mostly a factor of cost rather than technical feasibility. The reason for this is that the flexibility required of the systems is limited due to the mostly static or predictable nature of the surrounding. A key concept for mobile autonomous systems today is the ability to operate in environments, which are not particularly designed for them.

The mobility of the systems is a decisive factor, which has a large potential to have a significant impact on our everyday life. By not being limited to a single place, single or multi-robot systems can provide flexible services in situations that would be dangerous, uncomfortable or not economically viable for human

presence. The notion of dangerous, dull and dirty which is often used to describe the scenarios which should better be handled by robots misses the important factor of economy. Even if there are machines that can perform the work, companies will pick humans over machines as long as they provide the same quality and risk, for a lower cost. One important cost factor for machines is the human operator. The more machines can do autonomously, the more likely they are to become economically viable over time.

Obviously one of the most important abilities for a mobile robot is to navigate within its environment, in other words to get from point A to point B, ideally in the most effective way. From an engineering point of view, how difficult this is largely depends on the environment and the resources and abilities of the robot. In principle, a robot which could only perform actions without the ability to sense its environment would be able to get from point A to point B in finite time by performing random movements. Apart from the fact that this seems far from effective, the robot also would not know if it actually reached B. A more sensible approach seems to be to let the robot sense its environment. If the environment itself provides enough clues on B, either as an intrinsic property or because it was designed that way, the robot could be stateless. One example for this are RFID tags embedded into the floor. In this case, a model of the environment might not be necessary. In the most general case this can not be assumed, and a model is required to effectively get from A to B. Figure 1.1 shows the skills which are needed to perform navigation for this general case, and the intersection of those skills.

The generation of a model of the environment is called *mapping*. It generates associations of real world properties in a model of the world. This could for example be the matter density in a specific volume in space. The generation of the model requires that the system can sense its environment. The matter density for example can be sensed by how much light is permeating the volume, or by sensing the reaction forces on the robot.

To gain knowledge of the position of the robot within the surrounding is called *localisation*. Sometimes the position within the surrounding can be measured directly, but this is not the case in general. Performing *actions* is how the position of the robot changes in the environment. The action part – in the context

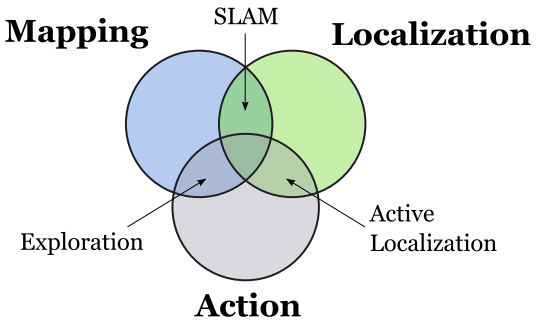


Figure 1.1: The skills required to perform navigation can be broken down into mapping, localization and action. This figure is adapted from Stachniss et al. [2004] and shows the overlapping areas of those skills.

of navigation – is a combination of decision-making and motion control of the actuators, and while it is very relevant to the subject at large, the focus of this work is on the localisation and mapping part in the context of body-environment interaction.

Both localisation and mapping can appear in isolation on a system. If a model of the environment is known, it might not be required to perform mapping to generate one. This is of course under the assumption that the environment does not change significantly. A good example is a car navigation system, where a model of the world is known in the form of a street map. The car receives measurements from the Global Positioning System (GPS), which are combined with the road-map, and the assumption that the car travels on a road. Based on that, actions are calculated, which are then communicated to the driver. In this case it is assumed that the driver can compensate for changes or uncertainty in the model, like road-works or traffic-lights. Conversely, a surveying robot might have a perfect localisation – or near perfect, given the task – but uncertain knowledge of the environment, for example in an rural or natural setting. In this case sensor information is used to build up a model of the environment, which is adequate for the task. For example if the task is just to navigate, the relevant information would be the traversability of certain areas in the environment. The traversability is a measure of how well a given system is able to traverse a certain terrain.

More often than not, neither a model nor a means of global localization is available to the system. One example for this are exploration robots in space

environments. They have neither a means for global localisation like GPS, nor a detailed map of the environment. Further, because of visibility and signal delays remote operations is often not practical, and operations benefit significantly from partial or fully autonomous behaviour. In such cases, the Simultaneous Localisation and Mapping (SLAM) problem has to be solved. One of the initial works on the subject of SLAM was performed by Smith and Cheeseman [1986], and the term itself was later coined by Durrant-Whyte et al. [1996]. Since then, the work on the SLAM subject has largely dominated the active research for autonomous robot navigation (Smith and Cheeseman [1986]; Smith et al. [1990]; Durrant-Whyte et al. [1996]; Thrun and Bücken [1996]; Hertzberg and Kirchner [1996]; Fox et al. [1999]; Montemerlo et al. [2002]; Bosse et al. [2004]; Thrun et al. [2005]; Frese [2006]; Thrun [2006]; Nüchter et al. [2007]; Grisetti et al. [2007]; Olson [2008]).

At the core of SLAM lies the data association problem. All SLAM methods work by identifying the same things from different locations. A small motivational example for this thesis is given in Figure 1.2. A robot perceives two stones at some point in time. He then moves towards the stones and senses the first stone trough contact at another point in time. By associating the two percepts he can make some useful inferences. He can estimate the distance he travelled. He can also assume that he is about to encounter a second stone, even though he does not see it at this point any longer. We will revisit this example throughout the introduction chapter to explain some of the relevant concepts for this thesis.

When implementing SLAM algorithms for real robotic systems, the concepts from the previous example are not so simple any longer. For example, how does the robot know that it really is the first stone, and not the second one? Noisy sensors, complex systems and dynamic environments can make the SLAM problem very difficult. The application of Bayesian statistics has proven especially useful in this context, as it provides a framework to model the uncertainties involved in real systems. These, uncertainties originate mainly from the fact that it is nearly impossible and usually not desirable to model all facets of the world, or sensor systems that measure aspects of the world. Deviations from the model are therefore considered noise, which is most of the time easier to handle than it is to refine the the model. The general idea is that the true state of the world, which

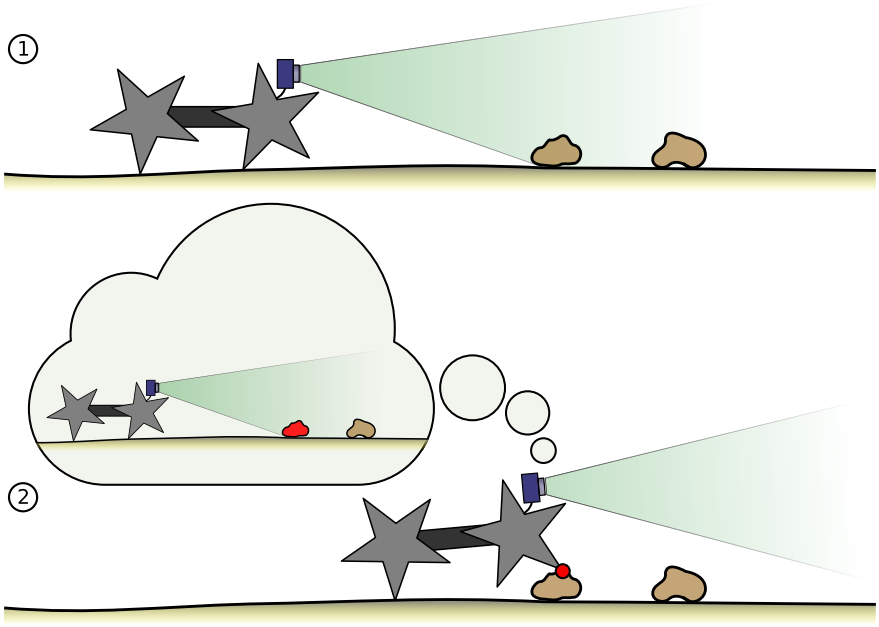


Figure 1.2: A robot senses two stones with the help of a vision system (1). At a later point in time it touches the first stone (2). By associating the two percepts (red) he gains useful knowledge about how far he has travelled in the past and that he is likely to encounter a second stone in the future.

includes the pose – the position and orientation – as well as the environment is only visible through noisy measurements of the world. Based on these measurements, a belief of the true value is formulated as a probability distribution over the possible values for that state. In general some initial belief is recursively updated by a *prediction* step, which uses a model of the system and sometimes the environment on how the system could develop in the near future. Often for example this is an ego-motion estimate, which may take the control inputs to the system, or the maximum applicable force values into account. It can then make a prediction on future probability distribution of the state. Measurements on the other hand can perform a *correction* of this belief, based on a model of the system and the sensor.

The motivational example from Figure 1.2 is sketched in Figure 1.3 formulated as a Bayesian filtering problem. The robot whose initial position is known with a degree of uncertainty visually senses some stones in the environment. The positions of the stones is not exact, so a belief is formulated on the existence of stones at certain positions (mapping). Now the robot moves a given distance.

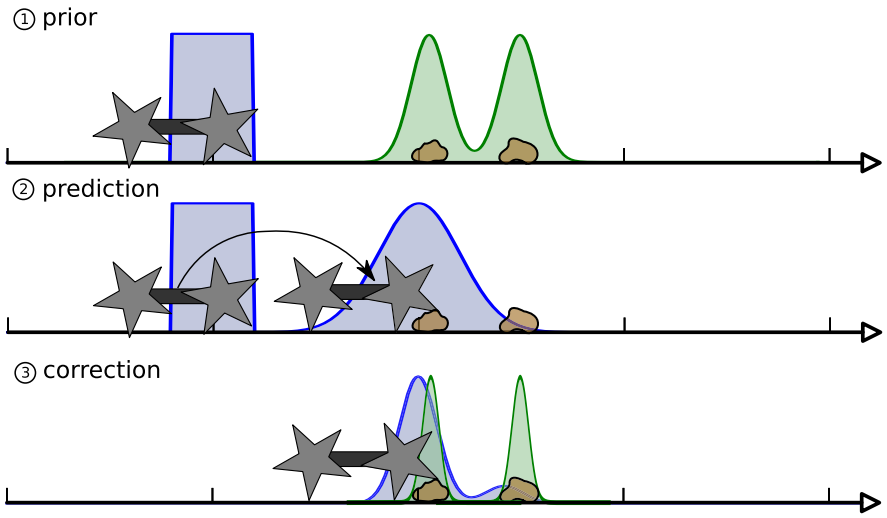


Figure 1.3: Bayesian statistics can be used to model the example given in Figure 1.2. The position of the robot is represented as a probability density function (blue), as is the distribution of stones in the environment (green). The prediction step increases the uncertainty on the position, while the correction step reduces both the position and the map error.

How much the robot has moved is also not exactly known, but based on counting actuator steps (odometry). Given a model, this allows to predict where the robot position is likely to be after it has moved. A stone is perceived by the robot by stepping on it in the next step. This measurement allows the correction of both the belief over the position of the robot and the position of stones in the environment. The problem is that there is no general closed form solution available to calculate these beliefs. In the general case all available methods to solve the SLAM problem perform some form of approximation or assumptions. Rao-Blackwellized Particle Filters (RBPF) and pose constraint graph optimisation (GraphSLAM) are two methods which have proved successful in the past. There are two main differences between these two methods. The first difference lies in the representation of the probability distributions. Particle filters use a sampling of the target distribution to approximate the probability density function. The second difference is that the particle filter performs calculations only on the current state and does not correct previous states. On the other hand, the pose graph based approaches form probabilistic constraints between poses at different time steps, and then try to find a constellation of poses (present and past) which minimizes the overall error.

Using particle filters for solving SLAM problems has been mainly introduced by Doucet et al. [2000a] and later integrated into the popular FastSLAM algorithm (Montemerlo et al. [2002]). The main principle of the particle filter is that instead of using a parametrized representation of the probability density function, like for example a Gaussian distribution, particle filters use a sampling from the target distribution to approximately represent this distribution. The particle density here should be proportional to the belief over the probability density function. Since there are only a finite number of particles, the sampling can only represent an approximation of the true probability density. In practice however, this can be an effective way to represent non-linear distributions with few state dimensions. The Rao-Blackwellized particle filter is a variant of the particle filter which allows to mix state representations, and is particularly suited for SLAM problems, where the large number of state dimensions for representing the environment would be prohibitive otherwise. The principle idea behind RBPF for SLAM is that the map is marginalized from the full state representation, and can be handled using other means of representation, while the robot position is still represented using sampling. Each of those pose samples (particles) holds its own map. Within the context of this map, the pose of the particle is considered the true pose. The particle filter reinforces the particles which are consistent with the sensor readings. In regular interval, the particles are resampled from the target distribution so that the particle density stays approximately proportional to the target probability density function. Particle filters are effective for highly non-linear belief functions over the state and at solving the data association problem.

There are two issues, which often arise when particle filters are used for larger scenarios. The first issue is that the resampling process works by effectively copying some particles and deleting others. Over time all particles will share a single strand of history. This property can be a problem when the likelihood of a particular history can only be evaluated after some time, for example when closing a large loop. The second issue is that when the pose uncertainty rises over time, the absolute particle density gets lower, since the particles have to cover a larger area. This *particle depletion* problem leads to situations where the particles are an insufficient approximation of the probability density function.

The graph based SLAM approach uses a different method for solving the SLAM problem. A popular version of pose graph optimisation based SLAM is the GraphSLAM (Thrun [2006]) method. Instead of filtering for the current pose and map, the GraphSLAM approach is a method which calculates the most likely manifestation of all past and present states taking all measurements into account. This is done by representing all poses of the system as well as optional landmarks of the environment as vertices in a graph. These vertices are connected by probabilistic pose constraints. For example the odometry in the prediction step forms a pose constraint between two consecutive poses. The next pose is related to the previous pose based on a noisy estimate on how much the robot has travelled. Landmarks can also be modelled as vertices, and the measurement of these landmarks forms edges between that landmark and the pose from which they were perceived. Instead of estimating the position of landmarks, an alternative way of using this filter is by rigidly attaching a local model of the environment to the poses and then performing matches between those local models in order to get constraints between the poses. Since pose graph based SLAM performs the optimisation over all pose nodes and not just the current pose, it is very well suited for generating globally consistent maps. Also, it is possible to perform explicit loop closing, something which is not possible with the particle filter. One of its problem is that the efficiency of the method degrades with larger number of vertices. For that reason it is more suited for larger time steps between poses and (if any) sparse representation of landmarks in the environment. A second issue is that of data association. Pose graph based methods do not handle wrong data associations very well.

In most real applications, sensor data needs to be pre-processed before it is submitted to the algorithm. So, actual SLAM implementations can generally be split into two parts: the front and the back-end. The front-end is system specific and includes measurement and update models as well as heuristics. It then generates constraints that are understood by the back-end. The back-end is usually very generic, and performs the Bayes specific calculations based on the chosen SLAM method. This split into two parts is especially visible in the GraphSLAM case, where all the data association and landmark extraction work is performed in the front-end. The diagram in Figure 1.4 shows the general

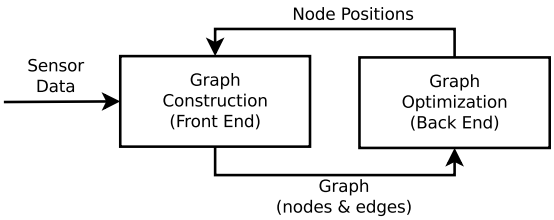


Figure 1.4: Practical SLAM implementations are often split into two separate parts. The front-end is system specific and performs the data associations based on the systen and measurement models. The back-end uses optimization methods to relax the constraints generated by the front-end. Often the front-end constitutes a substantial part of the overall SLAM solution (Olson [2008]). Figure adapted from Burgard et al. [2011].

relationship between the two parts for a GraphSLAM approach, but the general principle is also valid for the RBPF SLAM.

Most front-ends for SLAM solutions in robotics research use optical sensors like cameras or laser range finders to measure the environment. Robot poses are associated by correlating the current sensor percept with things that have been seen before. The robotic system is modelled as a point without spatial extents. In terms of localisation and mapping the only physical interaction of the robot with the environment which is taken into account is the odometry.

The motivation for this thesis is the belief that the interaction of the robot with the environment provides important information for localisation and mapping, which has not received enough attention. This information stands orthogonal to existing solutions in the sense that it exploits different modalities and as such offers the possibility to augment visual means of navigation.

In the area of robotics, the terms *proprioception* and *exteroception* have been adopted from the cognitive sciences. They classify perception into different categories. Exteroception are senses that are directed externally, so in human cognition terms for example vision and hearing. Proprioception is usually identified as the sense of self, so for example the knowledge of the limb positions. In cognition the term interoception is also used, and classifies senses of the internals of the human organism. In some cases the classification is not very clear even for human cognition. The sense of touch is connected to the body, but it is a matter of interpretation if the deformation of the body is the percept, or the originating

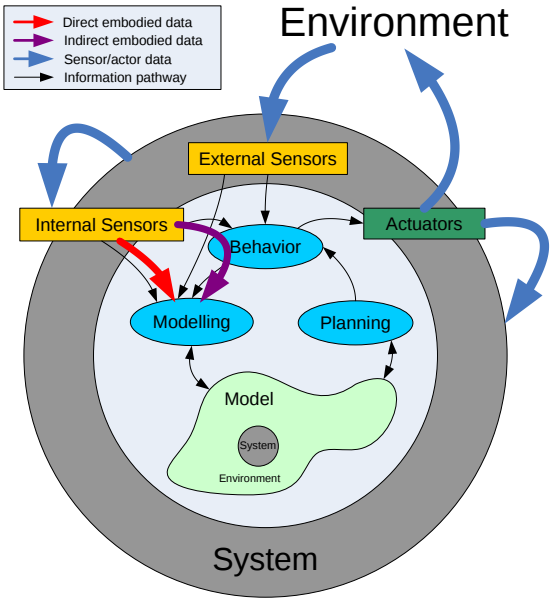


Figure 1.5: System model to describe the notion of embodied data on an informational pathway level. The illustration shows a generic description of a hybrid architecture (Arkin and Balch [2010]) reshaped to a system centric view.

external force. To avoid the confusion, we coined the term *embodied information* in Schwendner and Kirchner [2010], defined as sensory information originated within or on the border of the system in question.

Figure 1.5 shows an interpretation of a generic hybrid robotic system architecture with an emphasis of showing the separation between external and embodied information. These types of architectures take both the behaviour or reactive part and the planning or deliberative part into account, and combine them in a common framework. The information pathways in Figure 1.5 are such that they would fit any classical hybrid architecture (e.g. Arkin and Balch [2010]). There is a short behaviour loop, which takes external sensor information, processes it in a behaviour and generates actuator commands. These behaviours usually take direct actions, without generating a complex world model. For example the behaviour to follow a road might only require to sense the road-markings, but would not need a street-map. More complex actions take the longer loop, which also includes modelling and planning. In the autonomous car example this would include deciding, when to take turns and so on. For embodied data,

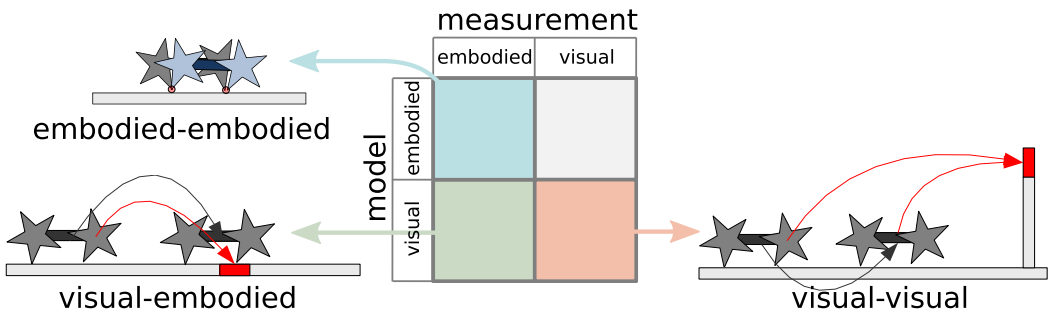


Figure 1.6: This data association matrix shows the different forms of associating a model with a measurement. Visual-visual association is the predominant method used in visual SLAM. The association of embodied with embodied data is used for the odometry model in Section 2.2. Visual-embodied correspondence is the primary association used in the measurement models in Chapter 2.

there are two information pathways in Figure 1.5 which can provide relevant information for an embodied SLAM (eSLAM) algorithm. The *direct* pathway starts at the internal sensors and is processed in the modelling task as is. This would include sensory information which can be directly sensed, so for example touch information or the orientation of the system with respect to gravity. This is very relevant information, and it forms a majority of what is being used in this thesis. There is another pathway which also provides relevant information. Behaviours usually require sensory input, which of course includes the internal or embodied information. Even though a behaviour might be model free, it can still have a state or some other type of processed information which is derived from the sensory input. This might for example be some indicator for a locomotion behaviour on how well it performs on the current terrain. This information can also be used in the modelling process. The information is *indirect* in the sense that it is routed through one or more external behaviours. Some further discussion on the subject is given in Schwendner and Kirchner [2010].

The fundamental property of how SLAM works is by performing data association. Perceiving the same things from different viewpoints is how a model of the surrounding and the position of the system can be extracted. One of the key challenges is to identify which of the perceived things actually originate from the same physical entity. In visual based SLAM systems, this is done by comparing one visual percept to another visual percept. By introducing embodied data, a

number of potential cross combinations for performing data associations arise. Figure 1.6 shows the association matrix for model-measurement associations when including visual and embodied data. Classical visual SLAM methods operate in the lower right section of the matrix only. The inclusion of embodied data offers three additional options for performing data associations.

embodied-visual includes data associations where the model – the previous measurement – stems from embodied information, and the current measurement is vision based. One scenario could for example be stepping on a stone, and afterwards performing visual inspection. This type of association is not covered in-depth within this work, but especially in the context of active perception (e.g. Stachniss et al. [2004]) it could prove an important source of information.

embodied-embodied does not include any source of visual information in the association, so both the model and the measurement come from embodied information. Especially systems with a high degree of tactile abilities (e.g. Aggarwal and Kampmann [2012]; Fox et al. [2012]) benefit the most from this type of data association. Even for systems with limited tactile abilities, this form of association can be used for example in the context of step counting for determining the relative pose change of a robot. Section 2.2 shows an odometry model which is based on this paradigm.

visual-embodied are those associations, where something was perceived visually (model) and then using the embodied information (measurement). An example again is seeing a stone, and then stepping on it, while it might already be outside the field of vision. This is actually a very relevant data association for navigation. For one, these types of associations are available in most system configurations, and secondly it provides navigation relevant information in the direct vicinity of the system. The navigation performance is directly connected to how well the system handles the environment. Visual-embodied data association have the highest information gain in the area which are most relevant for navigation: the current position of the robot. An example for what is meant by navigation relevant would be given by a

robot which has to pass a narrow bridge. A few centimetres of position error will make the difference between staying on the bridge and falling. If one of the actuators moves passed the edge, embodied information will directly provide this vital information, allowing the system to react immediately. The majority content of this thesis is concerned with visual-embodied data associations.

In this thesis, these additional data association methods are evaluated in the context of Bayesian methods for SLAM. A framework is generated, which allows the inclusion of this type of data in combination with more classical visual-visual correspondence paradigms. The eSLAM framework combines a Rao-Blackwell Particle Filter based approach for the generation of local maps, with the use of different measurement and update functions based on embodied data. These local maps are combined in a GraphSLAM-like fashion in order to include global constraints from different methods of data associations. The high-level contributions of this thesis can be formulated as follows:

- Formulation of how body-related data is highly relevant for navigation
- Development of the eSLAM framework for navigation relevant maps by including embodied information
- Novel combination of RBPF and GraphSLAM based approaches, which combines some of the advantages of each method
- Investigation on how this type of method integrates in a full navigation solution
- Evaluation of eSLAM on different systems and in different environments

1.2 Foundations

In this section, the methodological foundations of probabilistic SLAM are summarized. It includes some formal derivations, which are there to improve the understanding of the concepts. Parts of the formulation are later referenced in

the models Chapter 2. The derivations in this section are in large parts based on Bailey and Durrant-Whyte [2006], Thrun [2006] and Godsill [2009]. All the methods explained here are textbook material, and relate to the motivation given in the previous section. Readers with a strong background in the probabilistic SLAM foundations are advised to skip this section.

> "We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes." (Pierre-Simon Laplace, A Philosophical Essay on Probabilities, 1814)

Put into mathematical terms, this would mean for the finite state of the world $x = \mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_N$, if the initial condition x_0 at time 0 was known and we had a perfect model

$$x_{k+1} = f_p(x_k) \tag{1.1}$$

which could provide the state x_{k+1} at time $k+1$ based on the previous state x_k, the state could be calculated for any time. This intellect, which is sometimes referred to as Laplace's Demon, leaves no room for free will and has since been challenged by the concept of the inherently non-deterministic quantum world. Laplace's proposition is of course of philosophical rather than practical nature, but the principal method of using models to perform predictions on subsets of the state of the world is fundamental to most scientific disciplines. In the case where x is a subset of the world, (1.1) does not generally hold any longer, since parts of the world which are not modelled could influence the state. One method which is often used in this case is to postulate a closed world, which ignores these external influences or add a noise term which includes all external influences or non-deterministic behaviours, such that

$$x_{k+1} = f(x_k) + e_f \tag{1.2}$$

In both cases, even if the state x_k was somehow known, we can only predict an estimate $\hat{x}_k = x_k + e_f$ of the actual state. Moreover, knowing the state is something which rarely ever happens. Even the initial conditions x_0 are in practice either unknown or estimated. Sequential computations of (1.2) thus have to use the estimate $\hat{x_k}$ instead of the actual state x_k. The deviation of $\hat{x_k}$ from x_k will accumulate over time, such that the estimate will become increasingly meaningless unless it is grounded on observations of the actual state. Such measurements can be formulated as:

$$z_k = h(x_k) + e_h \tag{1.3}$$

The combination of (1.3) with (1.2) and how the uncertainty of the noise terms are handled are general problems of sensor fusion. Bayesian methods have been used quite successfully for this purpose.

The underlying principle in Bayesian inference is the notion of prior and posterior probability distributions over random variables. In this context we define a random variable X such that at any time k the variable has a value $X_k = x_k$ based on the probability distribution $p(X)$. The probability distribution gives a relative likelihood for a particular outcome of the random variable at time k. In Bayesian statistics, this distribution is called the prior, since it provides an information on the uncertainty of a particular state prior to additional knowledge. Additional knowledge can be given in the form of conditional probabilities. The distribution $p(X|Y)$ is the probability distribution over X given the probability distribution over Y.

Using the tools of Bayesian statistics, the prediction of the state (1.2) can now be reformulated as

$$x_k \sim f(x_k|x_{k-1}) \tag{1.4}$$

which states that the state x at time k is a draw from the probability distribution over x_k given the previous distribution x_{k-1}. The noise term e_f from (1.2) is contained in the probability distribution f, which assumes a Markov structure of the state.

$$f(x_k|x_{k-1}) = p(x_k|x_0, x_1, \ldots, x_{k-1}) \tag{1.5}$$

The Markov property states that all future conditional probabilities only depend on the current state and not any of the past events. This essentially means that the state x_k is memoryless and encodes all information that is necessary to model the state. In reality of course the Markov property is violated under a closed world assumption, because there are always events that may not be encoded in the state that can effect future states. In analogy to the prediction, the measurement equation (1.3) can be formulated as a stochastic process, such that

$$z_k \sim g(x_k|z_k) \tag{1.6}$$

where g is the distribution given by

$$g(x_k|z_k) = p(z_k|x_0, x_1, \ldots, x_k, z_0, z_1, \ldots, z_{k-1}) \tag{1.7}$$

1.2.1 General Bayes Filter

The distribution of $p(x_{0:k}|z_{0:k})$ provides the probability distribution over all the states up to k given all the measurements. Often one is only interested in the current state and can therefore revert to the more simple posterior distribution of

$$p(x_k|z_{0:k}). \tag{1.8}$$

This is often referred to as the filter or on-line formulation of the problem.

Solving this formulation can be performed recursively using $p(x_0)$ as a starting prior. Given such a prior distribution a projection step is performed, which gives the prior distribution $p(x_k|z_{0:k-1})$ before introducing the measurement z_k.

$$\begin{aligned} p(x_k|z_{0:k-1}) &= \int p(x_k, x_{k-1}|z_{0:k-1})dx_{k-1} && (1.9)\\ &= \int p(x_{k-1}|z_{0:k-1})p(x_k|x_{k-1}, z_{0:k-1})dx_{k-1} && (1.10)\\ &= \int p(x_{k-1}|z_{0:k-1})f(x_k|x_{k-1})dx_{k-1} && (1.11) \end{aligned}$$

Equation (1.9) is a formulation of the Chapman-Kolmogorov equation, which performs an elimination of the x_{k-1} term using marginalization. Because we are in

the continuous domain, this marginalization equates to an integral over x_{k-1}. The chain rule is used in (1.10) to split the joint probability into the prior probability $p(x_{k-1}|z_{0:k-1})$ and the prediction function f from (1.5) in equation (1.11). The correction or measurement step can be formulated based on this prior with the Bayes rule:

$$\begin{aligned} p(x_k|z_{0:k}) &= \frac{p(z_k|x_k, z_{0:k-1})p(x_k|z_{0:k-1})}{p(z_k|z_{0:k-1})} \\ &= \frac{g(z_k|x_k)p(x_k|z_{0:k-1})}{p(z_k|z_{0:k-1})} \end{aligned} \tag{1.12}$$

Together the equations (1.11) and (1.12) form the general Bayes filter formulations. They are the foundation of the two methods which are used in this work: the Rao-Blackwellized Particle Filter (RBPF) as well as the Pose Graph Optimization.

An example for the individual steps of the Bayes filter is given in Figure 1.7, which shows a simplified version of the general idea of this work. A mobile robot has a known distribution over the position, and perceives obstacles in the distance. Uncertainty in the movement prediction spreads the mass of the position distribution. In the measurement step one of the obstacles is directly perceived by the robot, which allows a correction of the position distribution.

The problem with the general Bayes filter is that there are no closed solutions for the integral in (1.11) and the product in (1.12) in general. There are however solutions for parametric distributions, like for example the normal distribution.

1.2.2 Kalman Filter

If we assume that the distributions for the prediction f and the update g functions are Gaussian so that

$$f(x_k|x_{k-1}) = \mathcal{N}(x_k|Ax_k, C) \tag{1.13}$$

$$g(z_k|x_k) = \mathcal{N}(z_k|Bx_k, D) \tag{1.14}$$

where $\mathcal{N}(x|\mu, Q)$ is the probability density function with a Gaussian distribution of mean μ and covariance Q. In this case, there is a solution to the Bayes filter formulation. The formulation from (1.13) and (1.14) can also be formulated as

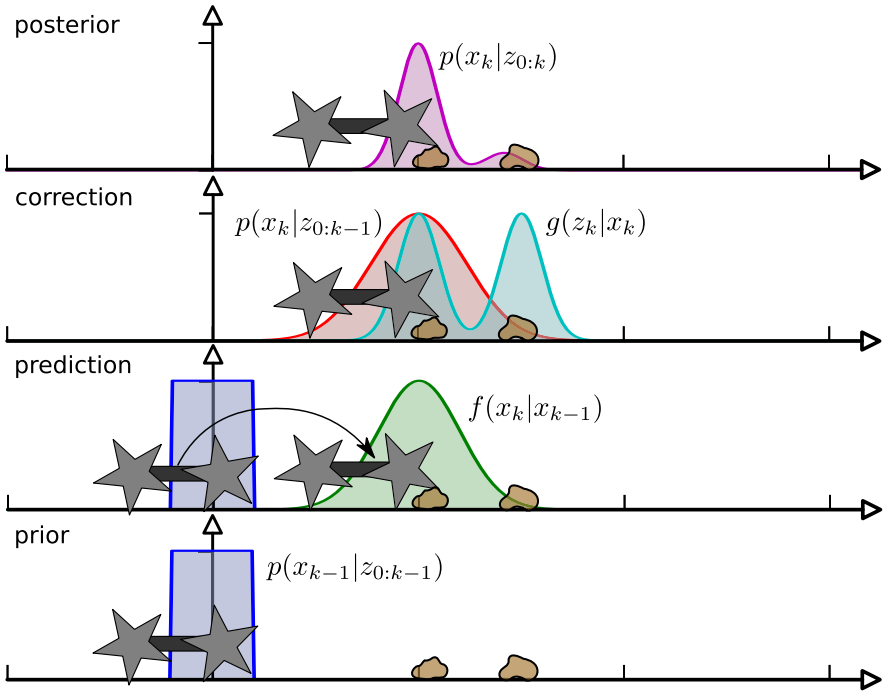


Figure 1.7: Shows an illustration of the steps involved in the general Bayes filter. The sequence starts at the bottom with the prior distribution (blue) over the position of the robot, and the perception of two stones. In the next step an odometry model is used to predict the distribution given the control inputs (green). The prior and forward prediction are convoluted to get the measurement prior (red). After the system senses a contact with an obstacle (cyan) shows the measurement likelihood given the position. The posterior distribution (purple) now includes all available information. Note that all distributions in this figure are normalized to their maximum value.

$$x_{k+1} = Ax_k + v_k \tag{1.15}$$
$$z_k = Bx_k + w_k \tag{1.16}$$

with v_k and w_k are zero mean Gaussians with covariance C and D respectively.

$$x_k \sim \mathcal{N}(x_k|\mu_k, P_k) \tag{1.17}$$

so that the state x_k is drawn from a normal distribution of mean μ_k and covariance P_k. Using the transformation of variables theory we can apply the transformation from (1.15) to the distribution given by (1.17), which leads to the new distribution

$$\begin{aligned} p(x_k|z_{0:k-1}) &= \mathcal{N}(x_k|\mu_{k|k-1}, P_{k|k-1}) \\ \mu_{k|k-1} &= A\mu_{k-1} \\ P_{k|k-1} &= C + AP_{k-1}A^T \end{aligned} \tag{1.18}$$

which gives the update steps for the Kalman filter. The measurement step can be derived from (1.12).

$$\begin{aligned} p(x_k|z_{0:k}) &\propto \mathcal{N}(z_k|Bx_k, D)\mathcal{N}(x_k|\mu_{k|k-1}, P_{k|k-1}) \\ &\propto \exp\left(-\frac{1}{2}([z_k - Bx_k]^T D^{-1}[z_k - Bx_k])\right) \\ &\quad \times \exp\left(-\frac{1}{2}([x_k - \mu_{k|k-1}]^T P_{k|k-1}^{-1}[x_k - \mu_{k|k-1}])\right) \\ &\propto \exp\left(-\frac{1}{2}([x_k - \mu_k]^T P_k^{-1}[x_k - \mu_k])\right) \\ &= \mathcal{N}(x_k|\mu_k, P_k) \qquad (1.19) \\ \mu_k &= P_k(B^T D^{-1} z_k + P_{k|k-1}^{-1}\mu_{k|k-1}) \qquad (1.20) \\ P_k &= (B^T D^{-1} B + P_{k|k-1}^{-1})^{-1} \qquad (1.21) \end{aligned}$$

using the matrix inversion lemma, this can be further simplified to the well

known Kalman update rule

$$K_k = P_{k|k-1}B^T(BP_{k|k-1}B^T + D)^{-1} \quad (1.22)$$

$$\mu_k = \mu_{k|k-1} + K_k(z_k - B\mu_{k|k-1}) \quad (1.23)$$

$$P_k = (I - K_kB)P_{k|k-1} \quad (1.24)$$

1.2.3 Sequential Monte Carlo Filter

If the assumption that f and g are Gaussian distributions can not be made, no general closed form solution exists. One effective way to overcome this problem is to use Monte Carlo methods. The general principle of Monte Carlo methods is to use samples to approximate some statistic over a distribution, or the distribution itself. The general expectation for an arbitrary function $h(x)$ over a distribution $p(x)$, where x is a continuous random variable can be given as an indicator function I_h.

$$I_h = \mathbb{E}_{h(x)}(p(x)) = \int h(x)p(x)dx \quad (1.25)$$

Now lets say that we represent the distribution $p(x)$ as a collection of N random samples $x^{(i)} \sim p(x)$ drawn from this distribution, where $i = 0, \ldots, N$. With the help of a Dirac delta function $\delta_a(b)$, which has the property that it integrates to 1 in the range $[a-\epsilon, a+\epsilon]$, an approximation of the indicator function can be given.

$$I_h \approx \hat{I}_h = \int h(x)\frac{1}{N}\sum_{i=1}^{N}\delta_{x^{(i)}}(x)dx = \frac{1}{N}\sum_{i=1}^{N}h(x^{(i)}) \quad (1.26)$$

$\hat{I}_h$ will approach I_h for large N. The function $h(x) = x$ for example would give the mean of the distribution. So it seems that for certain applications

$$p(x) \approx \frac{1}{N}\sum_{i=1}^{N}\delta_{x^{(i)}}(x), \quad (1.27)$$

which can be thought of as a sum of point mass distributions given by the Dirac function at the particle positions, is a useful approximation. The requirement for this approach to work, is that particles are sampled from the distribution $p(x)$.

Sampling from an arbitrary distribution is not always directly possible, even if the density function can be evaluated. One method to overcome this problem is to use importance sampling. In this case, a second distribution $q(x)$ – often called the importance distribution – is used for sampling, and then the individual samples are weighted so that they fit the target distribution $p(x)$.

$$I_h = \mathbb{E}_{h(x)}(p(x)) = \int h(x)p(x)dx = \int h(x)\frac{q(x)p(x)}{q(x)}dx \quad (1.28)$$

$$I_h \approx \hat{I}_h = \int h(x)\frac{p(x)}{q(x)}\frac{1}{N}\sum_{i=1}^{N}\delta_{x^{(i)}}(x)dx = \frac{1}{N}\sum_{i=1}^{N}\frac{p(x^{(i)})}{q(x^{(i)})}h(x^{(i)}) \quad (1.29)$$

Now, if we let $w^{(i)} \propto \frac{p(x^{(i)})}{q(x^{(i)})}$ we can reformulate the approximation of $p(x)$ where the particles are sampled from $q(x)$ as

$$p(x) \approx \sum_{i=1}^{N} w^{(i)}\delta_{x^{(i)}}(x). \quad (1.30)$$

The weights are defined as proportional to the ratio between the density functions $p(x)$ and $q(x)$, and need the additional constraint of

$$\sum_{i=1}^{N} w^{(i)} = 1 \quad (1.31)$$

so that the probability density (1.30) is properly defined.

With the ability to approximate $p(x)$ using a sampling from a different distribution, the general Bayes filter equations (1.11) and (1.12) can be approximated. The filter formulation are of course handled recursively in this case as well. A forward prediction is performed using the f density function. Then g is used to correct the prediction using the measurement.

Let us assume that we start with a sampling of the state distribution so that

$$x_k^{(i)} \sim p(x_k|z_{0:k}) \quad i = 1, \ldots, N. \quad (1.32)$$

We want to perform a forward prediction based on this, in order to get an

approximation of $p(x_{k+1}|z_{0:k+1})$. The problem is that we cannot directly sample from this distribution given our prediction and measurement distributions f and g. A common method – called the Bootstrap filter – directly uses the f as the importance distribution q, so that

$$x_{k+1}^{(i)} \sim f(x_{k+1}|x_k^{(i)}). \tag{1.33}$$

This gives an approximation of $p(x_{k+1}|z_{0:k})$ based on (1.11) and the fact that each sample was constructed based on a known previous sample. $p(x_{k+1}|z_{0:k})$ does not yet take the measurement into account. This is performed by adjusting the particle weights according to the importance weight.

$$\begin{aligned} w_{k+1}^{(i)} &\propto \frac{p(x_{k+1}|z_{0:k+1})}{q(x_{k+1})} && (1.34)\\ &\propto \frac{\frac{g(z_{k+1}|x_{k+1})p(x_{k+1}|z_{0:k})}{p(z_{k+1}|z_{0:k})}}{p(x_{k+1}|z_{0:k})} && (1.35)\\ &\propto w_k^{(i)} g(z_{k+1}|x_{k+1}) && (1.36) \end{aligned}$$

The importance weight in (1.34) is the ratio between the target distribution that we want to estimate – so the distribution of x_k including all the measurements – and the importance distribution. After substituting (1.12) and choosing f as the importance distribution q in (1.35), it becomes clear that choosing f simplifies the problem by cancelling out $p(x_{0:k}|z_{0:k})$. Also, the term $p(z_{k+1}|z_{0:k})$ is constant over x_k and can be removed in the proportional relation.

After the iteration step, the particles in combination with their weights now form an estimation of $p(x_{k+1}|z_{0:k+1})$, which is the required distribution. In principle, this is a sufficient estimation given a large enough N. The problem is however that after a while, the particle weights will start to diverge so that the influence of some of the particles on the indicator function are very small, up to the point where the entire indicator function is dominated by a single particle. This problem can be alleviated by performing a resampling step. Resampling will draw samples from the posterior distribution proportional to their importance weight. In this way particles that have a low weight have a higher chance to be substituted by particles with higher weight. One can think of this step as N available slots,

which are filled with particles according to their current importance weight. The resampling step will increase the variance of the indicator function, and thus should not be performed at every iteration step. Rather, the weights are updated according to

$$w_{k+1}^{(i)} \propto w_k^{(i)} g(z_{k+1}|x_{k+1}) \tag{1.37}$$

until the distribution of the weights has diverged too much from a uniform distribution. A good measure for this is the effective number of particles.

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \left(w_k^{(i)}\right)^2} \tag{1.38}$$

Once N_{eff} drops below a threshold value e.g. $N_{\text{eff}} < \frac{N}{2}$, the particles are drawn according to their importance weight. How the particles are drawn influences the variance of the indicator function. Ideally, some stratified methods are used, which prevent the resampling from generating sample distributions that are too far off the expectation (Hol et al. [2006]).

The particle filter is an effective tool to approximate the general Bayes filter using Monte Carlo methods. An example is given in Figure 1.8 which tracks different particles through the individual steps of the filter. The filter in its pristine form is very general as it does not make any assumptions on the state, measurement or update functions. This does not mean that it can be applied to all problems in practice. The main issue with the particle filter is that the quality of the approximation depends on the number of particles N. Especially when the state x_k is of higher dimension the size of N required to perform an appropriate approximation is often computationally prohibitive. Even in lower dimension a related problem can occur, which is often referred to as particle depletion. The particle density can become too low to properly approximate the shape of the probability density function, making the approximation of the posterior distribution insufficient. Even with many particles, this can happen, when the prior is flat and the measurement function g is highly non-linear in x_k.

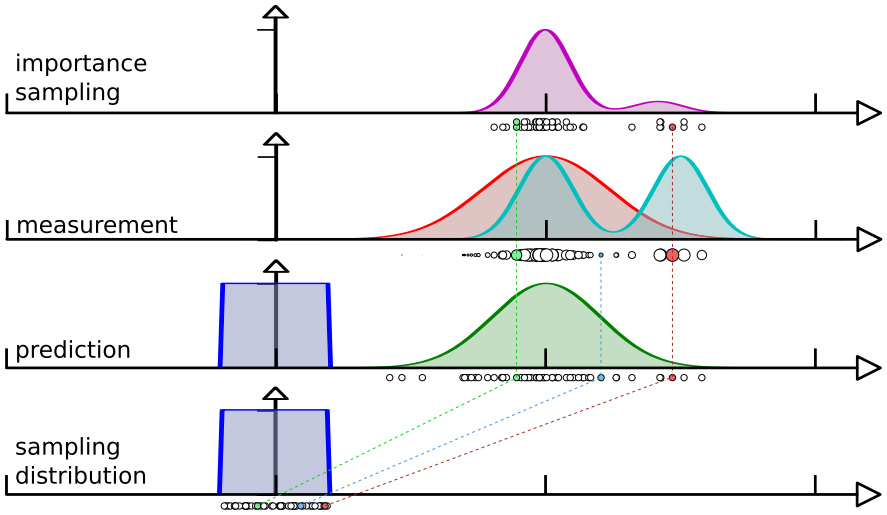


Figure 1.8: The particle filter is an approximation of the Bayes filter and uses particles for representing the probability distributions. The illustration shows how the example from Figure 1.7 would behave in a particle filter setting. Three of the 50 particles are highlighted. The red particle is carried forward to the next step based on the change in the prediction step. The low importance weight for the blue particle results in an elimination of the particle. The green particle has a high weight and is duplicated. The two green particles now share the same history.

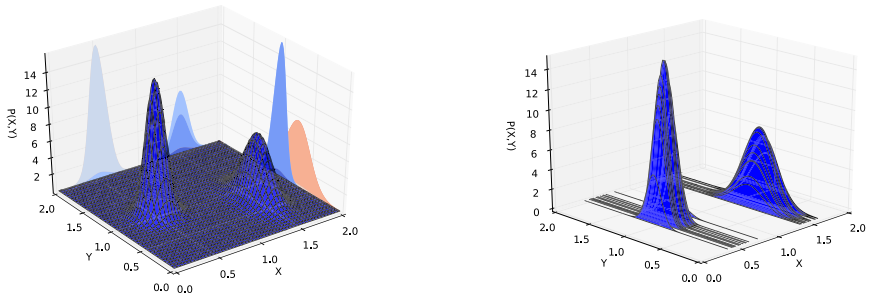


Figure 1.9: The joint distribution of X and Y (left) has multiple modes and would not fit a Kalman filter approach. A sampling representation of the distribution would require a large number of particles. The key feature of the RBPF is to exploit the Gaussian substructure and sample over the marginal of X. The conditional distributions $p(Y|X)$ for the samples (right) are well suited in this case to be approximated as Gaussians.

1.2.4 Rao-Blackwellized Particle Filter

We have already seen in Section 1.2.2 that the Bayes filter has a closed form solution if the assumption of linear state-transition and measurement functions are made and the prior is a Gaussian distribution. Especially for the field of robot localisation and mapping, this is an assumption that usually does not hold over the entire state space. However, often it is possible to model a subspace of x_k as Gaussian. For example, the position of the robot might require a non-linear model, while the velocity could be modelled in a linear fashion. Figure 1.9 shows an example of how such a joint probability could look like. For both random variables X and Y, the marginal probability is multi-modal and can not be represented using a simple parametrization. Of course a two dimensional Particle filter could approximate this structure, but would require a fair amount of particles for it. One can however note that the distribution of Y given X could be approximated by a Gaussian parametrization. The insight of the RBPF is to sample over the marginal of X and then use a Gaussian representation of the conditional probability. In this way only one dimension has to be sampled instead of two, and each sample contains a parametrization of the Gaussian. In many cases this makes the RBPF more effective when compared with a normal particle filter.

So lets assume that the state x_k can be divided into a linear part x_k^l and a non-linear part x_k^n. In this case the posterior can be factored into a linear part, which is conditioned on the non-linear part, and a non-linear part which is independent of the linear part.

$$p(x_k^n, x_k^l | z_{0:k}) = p(x_k^l | x_k^n, z_{0:k}) p(x_k^n | z_{0:k}) \tag{1.39}$$

Because of this, the filter is often called marginalisation filter. By approximating the non-linear part by a particle distribution, we can obtain the following equations:

$$\begin{aligned} p(x_k^n, x_k^l | z_{0:k}) &\approx p(x_k^l | x_k^n, z_{0:k}) \sum_{i=1}^{N} w^{(i)} \delta_{x_k^{n(i)}}(x_k^n) && (1.40) \\ &= \sum_{i=1}^{N} p(x_k^l | x_k^{n(i)}, z_{0:k}) w^{(i)} \delta_{x_k^{n(i)}}(x_k^n) && (1.41) \end{aligned}$$

The full probability distribution over the state is approximated by N distributions over the linear part. Since each of the linear distributions is conditioned on the $x_k^{n(i)}$, an update and correction step of the linear part is required for each particle. In the linear case this means, that there are N Kalman filters. Even with this overhead, RBPF perform better on states that can be split. The reason for that is the Rao-Blackwell theorem. It states that given an estimator $\sigma(x)$ for a random variable X the Rao-Blackwell estimator $\sigma_1(x) = \mathbb{E}(\sigma(x)|T(x))$ will be at least as good given that $T(x)$ is a sufficient statistics for X. The mean and variance suffices as a statistic of the Gaussian subspace in x, and hence the marginalized filter estimate is at least as good as the particle filter. This is no surprise given that we know that the Kalman filter is optimal for Gaussian subspace with linear update equations.

1.2.5 Graph optimization

The previous sections have assumed that the state of the world model x_k at time k is what we are interested in. Suppose now, that we would like to estimate the $p(x_{0:k}|z_{0:k})$, so the joint probability over the state for all time steps $0 \dots k$. Further, lets say that both the state update function f will relate the state x_{k+1}

with the state x_k with a linear model, and the measurement h does the same for the state x_i with an arbitrary previous state x_j, where $i \leq k$ and $j \leq k$. We can then interpret the information from both f and h to provide constraints between two states x_i and x_j. In addition, we make the assumption that the constraint between the states can be modelled with a Gaussian error, so that a measurement z_{ij}, relates two states by

$$z_{ij} = \hat{h}(x_i, x_j) + w_{ij} \tag{1.42}$$

where w_{ij} is a zero mean Gaussian noise with covariance matrix C_{ij}. The basic idea is that these constraints form a graph, where the vertices are the states, and the edges the constraints between the states. We can then formulate the problem so that we seek an estimate of the states $x^*_{0:k}$ such that the joint likelihood of all measurements is maximized.

$$\bar{x}^*_{0:k} = \arg\max_{x_{0:k}} \prod_{<i,j>\in\mathcal{Z}} p(z_{ij}|x_{0:k}) \tag{1.43}$$

Where $\mathcal{Z}$ is the set of edges for which a constraint exists. The fact that the constraints between the edges are Gaussian can be used here. Instead of the maximum likelihood, the log likelihood is minimized. By letting $e_{ij} = \hat{h}(x_i, x_j) - z_{ij}$ be the error for each constraint, perturbed by Gaussian noise of covariance C_{ij}, the estimate can be expressed as

$$x^*_{0:k} = \arg\min_{x_{0:k}} \sum_{<i,j>\in\mathcal{Z}} e_{ij}^T C_{ij}^{-1} e_{ij} \tag{1.44}$$

Note that the Mahalanobis distance is used here, which is a good approximation for the log likelihood of a Gaussian constraint (Blanco et al. [2012]).

The problem is non-linear because of the function $\hat{h}(x_i, x_j)$. It is however possible to linearise the equation, and use iterative optimization techniques. There are efficient solutions available to solve these types of problems, which exploit the sparse connectedness of the graph (Kümmerle et al. [2011]).

To understand the principles of the graph optimisation SLAM methods it is better to provide an example. Let's say the state of the robot x_k are 2D poses: position (x, y) and an orientation θ. A practical implementation would need to

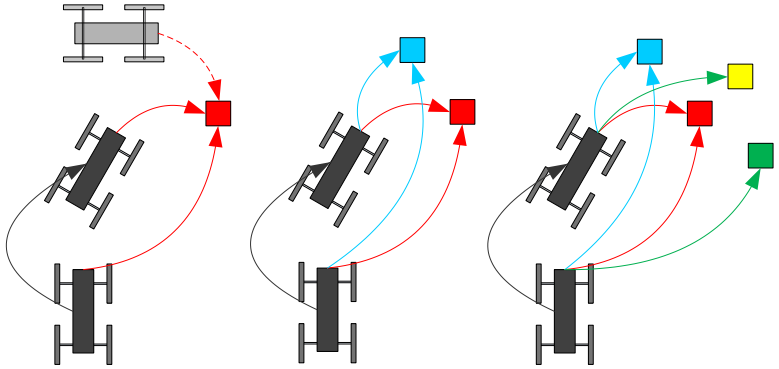


Figure 1.10: In the GraphSLAM approach measurements are used to perform associations between robot poses at different time steps. The association on the (left) is under-constrained in this example, and allows multiple equally likely pose changes. The (center) view is fully constrained and the pose change only has one possibility. Wrong data associations (right) are a problem for the Graph SLAM approach.

generate measurements z_{ij} between the poses at different time steps. This is usually done by associating environment features at pose i with features from pose j. Note that the features themselves can also be part of the optimization graph, as they measurement may also contain a large degree of uncertainty. In our example however, the feature positions are not measured separately, but are rigidly connected to the robot poses. The illustrations in Figure 1.10 shows how such measurements z_{ij} can be obtained. Measurement uncertainties, and the wrong feature associations as shown in Figure 1.10 on the right will lead to uncertainties with a covariance of C_{ij} of the pose measurement z_{ij}. As shown in Figure 1.11 on the left this will lead to an accumulation of the uncertainty with respect to the start pose over time. The advantage of the graph optimization based approaches are that any two poses can be associated with each other, and the error is distributed over all pose relations. So when the final pose in Figure 1.11 on the right is associated with the start pose again, the error with respect to the start pose is reduced for all nodes on the closed loop.

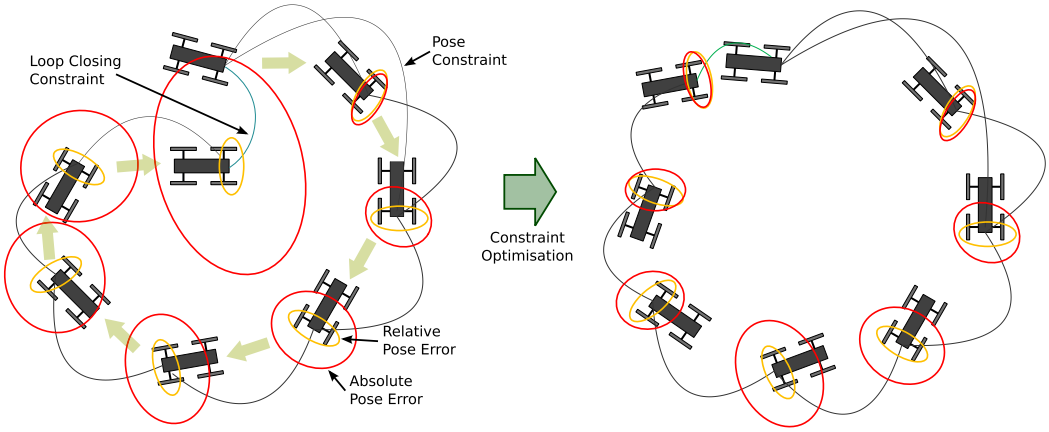


Figure 1.11: Pose associations are often formed in a linear fashion as shown on the left. The pose errors for each link (yellow) accumulate for the errors of each pose to the initial pose (red). By forming associations with poses that are already known (right) these errors can be reduced by distributing the error between the links.

1.3 Previous Works

The background and foundations Sections (1.1, 1.2) provided a general overview on the subject of probabilistic SLAM, and how embodied data can be used in combination with visual data for performing data associations tasks. The contributions of this work have been outlined at the end of Section 1.1 on Page 14. The subject of this section is how these contributions relate to the state of the art, and where some of the ideas for this thesis are originated. To address the different aspects of the contributions, the discussion of previous works is split into three subsections which closely relate to the contributions of this thesis. The first section is on the field of terrain aided navigation, which uses terrain features for localisation and mapping. In the second section an overview is given on how visual-embodied correspondence models are used in different contexts. Thirdly, the state of the art for hierarchical SLAM frameworks is presented.

1.3.1 Terrain Aided Navigation

One of the contributions of this work are the various measurement models for embodied-visual correspondence, and their application in localisation and SLAM

scenarios. An important related field, which was originated in research for cruise missile navigation is called *Terrain Aided Navigation*. The TERCOM system was originally envisioned in the 1950s by Chance-Vought (Golden [1980]) and later commercialized. The principle idea is that the cruise missile uses terrain elevation profiles along a preset flight path, which allows for correcting drifts of the inertial navigation system. A radar altimeter would measure the distance to the ground, and compare the relative change in elevation with the stored values. The SITAN system applied an Extended Kalman Filter (Hostetler and Andreas [1983]) to the same problem. Both methods require that the deviations from the predetermined flight path are small. An alternative approach uses a Point-Mass filter (Bergman et al. [1999]), which is much more flexible with respect to the kind of noise and prior distributions of the model. That work was later extended to a terrain tracking framework based on particle filtering (Gustafsson et al. [2002]). All these applications used a single distance measurement to the terrain and a known elevation model of the terrain. In underwater applications a similar setup – also known as bathymetric navigation – has become even more relevant (Carreno et al. [2010]), since there is no GPS available in underwater applications. The same holds true for space applications, where terrain aided navigation has been applied to e.g. precision landing (Trawny et al. [2007]). For ground based vehicles, there are many examples for terrain based navigation. Effectively everything that uses a terrain map for navigation qualifies as such. The utilization of street-maps in car-navigation systems are one of such examples (Gustafsson et al. [2002]). Here it is assumed that cars only travel on roads, and can thus be used as localisation constraints. An overview of methods for terrain based navigation and other methods for agricultural vehicles is given in Hague et al. [2000]. The methods mentioned in the work from Hague et al. emphasise on visual landmark recognition, which classifies as visual-visual correspondence in the terminology developed in Section 1.1.

The underlying idea of Gustafsson et al. [2002] and others is to use a particle filter for tracking the x and y direction and a marginalized Kalman filter for matching elevation readings is prominent in many related works. It is also the basis of the localisation method used in this thesis. The difference lies mainly in the measurement and update models, which use visual-embodied and embodied-

embodied data associations. Extending the pure localisation aspect of the RBPF is in principle straight-forward. Instead of just matching sensor data to a single map, each particle carries its own map, which gets updated with the sensor information. For 2D models there are many examples on how to implement this efficiently (e.g. Stachniss et al. [2004], Grisetti et al. [2005]). When performing this on 3D-elevation maps, RBPFs are more costly since the map update and matching functions have to be processed for each particle. An interesting example for an efficient RBPF implementation for elevation maps is presented by Barkby et al. [2009]. Instead of storing one map per particle, a single grid map holds all the measurements from all particles. This saves the cost of copying the maps when performing resampling. The method for generating local maps which is proposed in this thesis uses an approximated representation which stores when a map cell was last updated. The extended MLS used for the data representation is based on the principle idea by Pfaff et al. [2007] and the Gamma-SLAM approach (Marks et al. [2009]). Gamma-SLAM is aimed at estimating the variance of elevation in a cell instead of mean elevation. This is especially usefull for unstructured outdoor environments.

1.3.2 Visual-embodied data association

The application of non-visual correspondences in terrain based applications is sparse. The work presented in Chitta et al. [2007] uses a particle filter based approach to localize a four-legged robot using proprioceptive data. This is performed by calculating the environment contact points relative to the position of center of gravity for the four legged LittleDog robot (dimensions are 30 cm x 18 cm x 15 cm) and comparing it to a known elevation model of the environment. The contact points are based on the sensor values from the joint encoders and an AHRS. Using a simple measurement model, the filter was able to significantly improve localisation over odometry on a 120 cm by 60 cm terrain board. The contact model presented in Section 2.3 is similar to the work from Chitta et al. but extends to a probabilistic environment model and an estimation of the height-value using a Kalman Filter approach. A similar work has been performed by Dogar et al. [2010] in the area of mobile manipulators, where they augment

laser range finder data with proprioceptive and contact information with the environment to localize a mobile system in a household environment. A recent study by Fox et al. [2012] performs grid based SLAM using a whisker as the only sensor for generating a map of a simple box environment. Some very interesting work by Hoffman and Krotkov [1993] uses the legged Ambler system to map an unstructured environment. Association of visual and foot contact information is used here in a feedback loop to adjust the map height. Unlike the method proposed in Section 3.2 on local map generation, their approach does not take uncertainty into account. One example where visual-embodied association was actually embedded in a SLAM work is presented in Davison and Kita [2001]. In their work in an indoor environment, the roll and pitch sensor measurements of a wheeled system are used to augment a visual feature based SLAM method. In the work from Wurm et al. [2009] terrain classification based on the vibration of a wheeled system is used to train a classifier for vegetation versus non-vegetation using laser range finder remission values. That data association was not actually used for the navigation part, but for training a classifier of laser remission values. So, while there have been occurrences of using visual to non-visual association in the past, large parts of it is not actually used for navigation. Another example is the compelling work by Helmick et al. [2009] and Brooks and Iagnemma [2012], which use visual to non-visual association to classify the traversability of a terrain. This is effectively the reverse of how such a form of data association is used in my thesis.

1.3.3 Hierarchical SLAM

Although not strictly connected to the embodied correspondence model, as part of this thesis a framework for generating globally consistent maps suitable for navigation is developed. In the introduction, the general paradigms of RBPF and its limitations as well as the graph based SLAM were discussed. The idea of combining multiple levels of hierarchy into a SLAM process is not new. Thrun and Bücken [1996] generates an occupancy grid based metric map and calculates a topological graph consistent with this map based on Voronoi diagrams. The resulting topological graph is used for efficient path planning, and only a single

frame of reference is used for the complete map. The Atlas framework Bosse et al. [2004] uses a hierarchical representation which holds a topological graph over metric submaps. The submaps are generated using feature based EKF SLAM. A similar approach is presented by Estrada et al. [2005] adding the maintenance of loop consistency on the topological graph. The work by Blanco et al. [2008] also uses local metric submaps, which are generated using RBPF. The local map segments in this approach are connected by Gaussian constraints. The particle filter approach from Fairfield et al. [2010] is based on the same principles. The main critical idea here is that particles do not generate the entire history, but can be recombined over the segments to generate permutations. This is performed on a stochastic basis and no higher level constraints are used. The idea of recombining the particle trajectories from the local maps was also used in this thesis. Contrary to Fairfield et al. [2010] global constraints are managed using a pose graph optimizer. While Blanco et al. [2008] and Fairfield et al. [2010] remain within the realm of the particle filter in order to combine individual segments and solve the topological structure problem, the approach presented in this work generates isolated map segments that are combined in a pose constraint graph network. The advantage in my opinion is in the integration of arbitrary global constraints and the ability to use pose graph optimization frameworks, which have favourable global smoothing properties over particle filters.

Most of the hybrid SLAM approaches Blanco et al. [2008]; Bosse et al. [2004]; Thrun and Bücken [1996] operate on a 2D model of the environment, which simplifies the problem and has less computational requirements. This approach usually works well for operating an autonomous system in indoor environments where a certain structure and a flat ground can be assumed. For operating in complex outdoor environments this is often not sufficient, and other submap based approaches like Fairfield et al. [2010]; Nüchter et al. [2007] fill this gap by generating 3D models. Our method also operates on a 3D environment model, but uses assumptions valid for ground vehicles to generate smooth transitions between submaps. This property is important for using the maps in other parts of the navigation subsystem, like for example in a path-planning step.

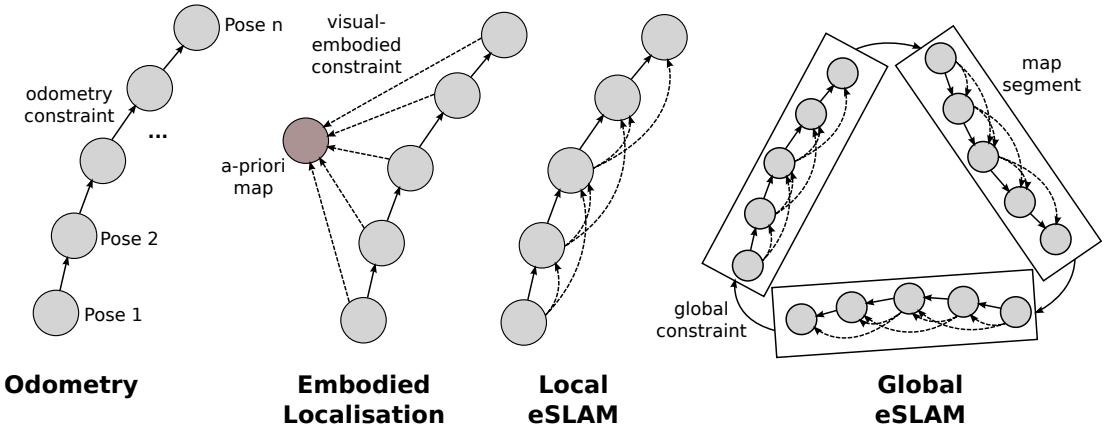


Figure 1.12: Multiple levels of constraints between poses are used in this work for localisation and mapping.

1.4 Method Overview

The method which is described in this thesis has the goal of generating navigation-relevant 3D models of the environment to be used on-line as part of a mobile robot navigation subsystem. A defining element is the use of embodied information for the navigation task. Different methods from the area of Dynamic Bayesian Networks are combined in a novel way, to model and process visual-embodied, embodied-embodied and visual-visual data associations. The nature of the body-environment interaction makes this approach mainly applicable to ground based vehicles with strong environment interaction.

1.4.1 Data Associations

There are multiple levels at which the different data association methods can be applied. An overview of the scenarios which are covered in this thesis is illustrated in Figure 1.12. The association of body-environment contact points is used in Section 2.2 to develop an *odometry* model. This model for dead-reckoning localisation serves as the basis for all embodied-embodied associations in the other configurations. The localisation error for dead-reckoning is unbounded without the integration of global reference measurements.

When there is an a-priori map available, visual-embodied data associations can be used in an *embodied localisation* scenario to provide such reference measure-

ments. We offer three different methods for performing these visual-embodied data associations. The contact model is described in Section 2.3, the slope based model in Section 2.4, and the terrain classification based model in Section 6.3.2. All three models connect a local embodied percept to an environment model. Multi-level surface maps (MLS) are used in our approach as the environment model for both localisation and mapping and are explained in more detail in Section 2.1. How these models are linked in a particle filter to perform localisation in a-priori maps is described in Section 3.1. The particle filter is particularly suited for this task as the visual-embodied map models are highly non-linear and have discontinuities because of the discrete map model.

The third association method from Figure 1.12 shows a configuration without an a-priori map. The filter setup for this *local eSLAM* configuration is detailed in Section 3.2. Instead of a global a-priori map, each particle carries its own map, which is updated by the visual sensors on the system. Visual-embodied associations are used here to link an embodied percept to a visual percept that has been made by the system a few time-steps in the past. The maps generated in this way are locally smooth, but are still prone to an unbounded drift in localisation error without referencing to a global map or revisiting previously seen places (loop closing).

1.4.2 Thesis Structure

The work can be split up into different parts, which is also reflected in the structure of this document. The description of the work is given in a bottom-up manner. First the individual models which are related to the data associations are developed. These models are then used in a particle filter based approach to generate local map segments. Local map segments in this context are small parts of the full map, for which the accumulated errors from the process are still acceptable for navigation. These maps segments are then combined using a global constraint graph: pose constraints on the map segments are combined in a graph structure and globally relaxed. A full map is finally extracted from this structure to be used in a navigation stack for an autonomous outdoor robot. The individual chapters of this document can be summarized as:

Modelling in Chapter 2 includes a description of the individual models developed in this thesis. One important part is the modelling of the environment. A unified structure for representing heterogeneous environments in a common data structure is presented. The individual map segments are represented in a grid indexed structure based on a modified version of the Multi Level Surface Map (MLS) representation. An odometry model which predicts the relative movement of the robot based on embodied-embodied data associations is developed. Further, three different models are given for visual-embodied data association. (1) A contact based model that associates the body-environment contact points of the vehicle with an environment model. This model is suitable for high resolution maps, where small terrain features can be distinguished. (2) The slope based model works for maps where the map sampling density is larger than the size of the vehicle. (3) For terrains that have different physical distinction, but different ground characteristics a terrain type classification model is proposed. Here visual-embodied association is performed based on visual and slip based terrain classification.

Local map segments in Chapter 3 are generated using a Rao-Blackwellized Particle Filter based on the models from Chapter 2. The embodied-embodied odometry model is used for prediction, and the visual-embodied models for the measurement steps. The measurement models can be used individually or in combination, based on the application scenario. The filter samples over the plane perpendicular to the gravity vector, and each particle in the filter generates its own 3D map using the environment model. The part which is parallel to the gravity vector – effectively, the height value of robot and map cells – is handled in each map using a Gaussian parametrization of the distribution. The maps which are generated using this method accumulate errors over time. Even though loop closing is possible using just the particle filter, the problem is that after a while all particles will – due to resampling – share a significant part of their map history. This means that different hypothesis in the past, which were equally likely can not be considered any longer. Also, in the context of the particle filter, it is more difficult to

include single global constraints based for example on visual-visual matching of feature points or ICP matching of point clouds.

Global map constraints in Chapter 4 tackle the mentioned issues of the particle filter, by generating smaller map segments instead of the full map. Effectively the state of the particle filter is stored at distinctive points in time. This includes all the maps in the filter as well as parts of the trajectories – so the past positions – of the individual particles. Consecutive segments are related by the final position of the previous segment, which is the starting position of the next segment. Additional constraints between the segments can be added for example to perform loop-closing. A method is described, which generates these constraints based on visual-visual data associations. After performing an optimisation of the constraint graph, a final map is generated. This is done by selecting one submap for each segment, which generates the smoothest transition between consecutive segments, by back-propagating the errors between the segments.

Navigation stack integration in Chapter 5 covers the integration of the localisation and mapping process into a navigation stack for mobile autonomous ground vehicles. The general problem here is that the map generation process is prone to change previous parts of the map continuously. When new constraints are added, the relaxation of the map might lead to a different arrangement of the segments. Any plan which is generated would have to be adapted to the new map, every time the map is updated. Since this is a costly operation, a different solution was opted for in this thesis. Two instances of eSLAM are running on the final system. One which performs the continuous mapping, and another which just does localisation based on the model, which was used for the last path planning step.

An overview of the approach, which also shows the sections where the parts are described in more detail is given in Figure 1.13. The structure is divided into three functional sets: localisation, local SLAM and global SLAM.

A description of the in depth experimental evaluation of the method is given in Chapter 6. Two different systems have been used for evaluation. The Asguard

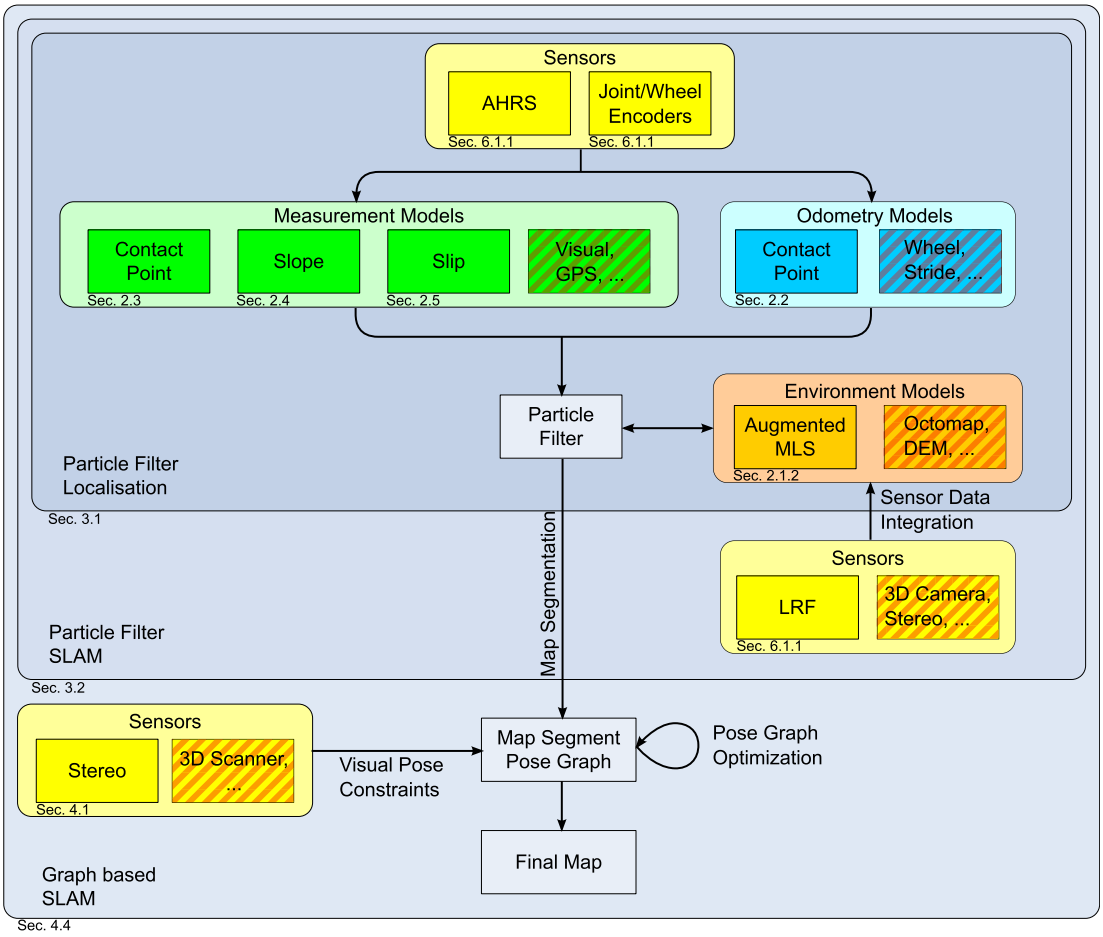


Figure 1.13: Shows an overview of the approach and the individual sections of this work where the methods are explained in detail. The hatched boxes mark parts which could also be used in addition or as an alternative, but are not part of the method description.

is a leg-wheel hybrid system, which has been used for most parts of this work. Additional results are also shown for the SpaceClimber, which is a six-legged walking robot.

Due to the suitability for using embodied data within the proposed framework the proposed method is termed *eSLAM* which is an acronym for embodied Simultaneous Localisation and Mapping. The method is not monolithic, and the parts described throughout this work are also valid in other contexts. The measurement, prediction and environment models can be used with other types of Bayesian filters. The particle filter for real-time generation of 3D environment models is suited to be used with other types of measurement models, as is the hierarchical method using the map segments in a constraint graph. In this sense eSLAM can be considered a set of tools which is particularly suited to incorporating embodied data in a SLAM process. Of course the proposed method is no exclusive solution for embodied data association. The general characteristics of the SLAM problem, which is spatial data association could be solved in various other forms.

1.4.3 Contributions

The individual parts and the combination of methods form the contributions of the eSLAM method to the state of the art. The following is a list of technical contributions:

- a variation of the Multi Level Surface Map (MLS) which uses a different interpretation of map updates and handles terrain slope properly.
- a contact based odometry model which uses the embodied-embodied data association paradigm to predict the relative pose change of a ground system with discrete body-environment contact points.
- a contact point based measurement model, which provides the likelihood of a set of contact points for a given position in the environment model.
- a scheme to combine multiple candidate contact points for the contact based measurement model without explicit contact information into a maximum likelihood estimate of the contact group.

- a slope based measurement model, which evaluates the likelihood of a series of contact point sets given a low resolution map of the terrain which includes slope information.
- a terrain characterisation based measurement model, which performs association of different terrain classes in both the embodied and the visual domain.
- a Rao-Blackwellized Particle Filter which incorporates visual-embodied and embodied-embodied measurement models and operates in 3D environment models. This filter can be used to perform localisation when given an environment model, or use sensor measurements to also generate a map. The contribution here is the integrated visual-embodied data association to estimate the distribution of the map and the robot pose over the spatial component parallel to the gravity vector.
- a graph structure that holds the state of multiple local filter states and encodes pose constraints between those segments of the full map.
- a method to generate a combined global map with optimal smoothing between map segments.
- a formulation of how the map generation process can be used on-line for a mobile autonomous outdoor system.

Chapter 2

Embodied Prediction & Measurement Models

This chapter covers the modelling aspects related to embodied data. Part of the motivation of this thesis was to generate maps of the environment, which are relevant for robot navigation. So an important factor is how these maps should be represented. This question is covered in Section 2.1.

Any Bayes filter has a prediction and a correction step. The prediction step requires a model which relates the previous and the current system state. For ground based mobile robots, an odometry model is what usually provides a prediction of the relative movement of the system. The contact based odometry model described in Section 2.2 is suitable for legged and leg-wheel hybrid systems, and is an example of embodied-embodied data association. The correction step uses a model to relate the current system state with a measurement. In this chapter, three different visual-embodied data association models are provided. The shape based contact model in Section 2.3 is used to estimate the likelihood for a contact point configuration for a specific position on a map. The slope based model from Section 2.4 provides an alternative method for large scale and low resolution maps. For terrains that have low variation in spatial features the terrain classification model from Section 2.5 can be used. That model is also an example for the indirect type, where the data for the model comes from a behaviour instead of a sensor.

The models in this section provide the heart of the embodied SLAM approach, and are used as the basis for the localisation and mapping methods presented in the following chapters.

2.1 Environment Model

Under the paradigm that the world is its own best model, behaviour based robotics has looked at the possibilities of performing robot control without the explicit generation of environment models. While this has been a successful approach for a number of use cases like for example the navigation of an office environment by the robot Shakey (Brooks [1991]), there are limits to the behaviour based paradigm. Complex tasks for both navigation or object manipulation often require the system to perform reasoning on the environment, and eliminate routes that are likely not successful or too costly. Physically enacting all these possibilities is costly at most, and often not possible at all. In order to perform reasoning and find a sufficient sequence of actions to achieve a desired goal, models are needed.

Most goals of a mobile robot are related to its environment. For example one such task could be for a planetary exploration system to go to a specific location, take sample below a rock and return the sample to the base for further analysis (Haarmann et al. [2012]). One additional constraint would be the requirement for the sample to be kept at a certain temperature by avoiding exposure by the sun. While one could conceive a behaviour based system which in principle would be able to solve such a problem, a more effective method is to generate a model of the environment and use it to perform reasoning. The shape of the terrain is estimated based on the sensor readings of the system. This model of the terrain shape is then used to calculate if the terrain is traversable or not, and a path is planned, which takes the robot from the current position to the goal. Taking the sample requires a combination of the actions of the mobile part of the system with the manipulation part for taking the sample. In order to prevent the sample to be overheated on the way back, the terrain shape and the information on the angle of the sun is used to estimate the path and timing for traversing the path which keeps exposure to a minimum. A critical element is the granularity at which the plan should be generated. Most of the time it is not very useful to

plan every action of the actuators over a prolonged period. This is very costly, and also does not incorporate situation changes and updates very well. A more effective method is to generate rough plans, for example the general direction, or the corridor (Joyeux et al. [2011]) and use a behaviour based approach for the actual execution. A combination of the planning and behaviour based approaches are generally referred to as hybrid architectures. The general information flow in such architectures is shown in Figure 1.5. A model should be generated at the level of granularity that is required for the planning to be successful. Often in hybrid architectures the requirements on the model are not as strict, since model inaccuracies can be handled by the behaviours once they actually physically encounter the situation.

A model can also incorporate other information about the environment such as colour, reflective properties of the material, material softness and so on. In all cases, the properties which are stored can only ever be estimations of the real properties and locations. A model of the environment is usually some spatially indexed structure which estimates property of the matter within a given volume of space. A very common type of model is a structure which divides the space into a regular grid structure in either two or three dimensions. Each element of the grid structure corresponds to a volume of space in the real world. For a mobile system it is usually important to see if it can physically penetrate this volume. An estimate for this property can be given by measuring if light passes through this volume or if it is reflected. In case it is reflected a simple estimator would assume that the system can not pass through this volume. On the other hand, when light passes through, such an estimator would give the information the system could also be situated in this volume. Such simple estimators are not always correct in the real world. A snow flake will readily reflect light, but would not stop a robot to get through. On the other hand, a glass door lets light pass through, but provides a physical barrier for a robot.

These examples are to emphasise the fact that environment models are always only estimates of some facets of the real environment, and can deviate from the ideal in many ways. Also, the model needs to be focused for the system and the task. A ground based system for example may not need to model the branches in a tree it passes by. It might however want to generate a model of the mountain

peaks in the far distance and use it for estimating its position.

2.1.1 Grid Based Maps

A common method for representing models of the environment is to use grid maps. A grid map generates a regular partition of space, where each grid cell represents a volume in real space. The biggest advantage of grid cells is that the regular structure forms an index over the space, and the data structure of the cells can be accessed in constant time given any position which is represented by the map. For indoor robots there are certain assumptions that can be made about the environment, which allows the system to use a 2D model of the environment. Each grid cell represents the volume of space given by a rectangular area on the floor extruded up to the height of the robot plus a margin. This is possible since the floor can be assumed flat. A simple model type would be if a point within such a volume is sensed as being occupied, the cell is considered non-traversable for the robot. This does not cater for uncertainties in the measurements. For that reason a popular extension which is widely used in practice are occupancy grid maps (Elfes [1989]). Occupancy maps contain a measure of belief on the occupation of a grid cell. These maps are very effective as they encode what is relevant for the navigation task of the robot. A path plan can for example take the belief on the occupancy into account and prefer areas which are more likely to be free. One problem of these types of maps is with the limitation to environments where an assumption for a fixed ground plane can be made. This is generally the case for building floors, but already fails for staircases. Multi story buildings can be handled by using multiple occupancy grid maps, but not by a single one, and stair-cases need to be handled in a special way.

For outdoor systems, especially in unstructured environments where the ground plane assumption is invalid, 2D environment models are not sufficient any longer. The fact that something that occupied space was detected by a sensor does no longer imply that the associated volume is not traversable and vice versa. What is considered the ground level at the point from which the percept is made may be completely different to the ground level of the perceived area. In order to infer whether a volume of space is traversable or not, the local surrounding has to

be taken into account, and based on that a traversable surface identified. One way of generating a 3D model of the environment is by extending the occupancy grid approach into the third domain. Where before a grid cell would represent a single volume of space, this space gets discretised in the z-axis. Such voxel maps have two major problems. One such problem is that of memory consumption. The memory requirements of the maps is multiplied by the number of vertical subdivisions of the cells, when using dense storage of the maps. There is a way around this problem by using hierarchical subdivisions of the entire map volume, like for example the Octomap (Wurm et al. [2010]) does. Memory consumption is reduced to acceptable levels in this case. The second problem of voxel based maps still persists, however. The discretisation of the space is necessary for a finite representation in memory, and generates aliasing effects. The differences in floor height are limited to discrete multiples of the voxel size. This puts an upper boundary on the voxel size in order to still be able to differentiate a traversable terrain – like a sloped floor – from a non-traversable terrain like the trunk of a tree.

Another very popular map model are the Digital Elevation Maps (DEM). This type of approach is often referred to as 2.5D maps, where the terrain elevation is considered a fractional dimension. This fractional dimension expresses the fact that the maps extend into the 3D space, but can not express arbitrary 3D surface structures. The model works in such a way that – like in the occupancy grid case – each cell represents a box volume which covers the full height interval of the map. For this volume the largest height value for measurements of occupied space is stored. This is acceptable for environments where the terrain height can be expressed as a single valued function over the x and y coordinates. This means that the volume of space above the ground needs to be non-occupied. Such a constraint is acceptable for example on flat pieces of the lunar surface, which do not contain any sort of cliffs, caves or similar structures. For terrestrial applications most type of vegetation like trees or structures like doors will violate this constraint and make DEMs hard to use without additional effort.

The general structure which forms the basis of the environment models used in this thesis is the Multi Layer Surface (MLS) map given in Triebel et al. [2006]. Instead of using a regular subdivision of the space of a grid cell, like the voxel

based approach does, MLS holds a variable number of surface patches in each cell. Each surface patch represents a non-overlapping height interval for which a measurement of occupied space has been made. In the original formulation of the MLS there is a distinction between horizontal and vertical surface patches. Each patch has a mean surface vertical position (z-value), and an uncertainty of the z-position in the form of its standard deviation. Vertical patches also contain the information on the height interval of the patch. In this way, horizontal patches represent to the top of a surface which could generally be traversable and solve the variable floor height problem mentioned earlier. Vertical patches on the other hand can represent walls, cliffs, tree trunks or any other mostly vertical elements of the environment. The MLS structure in this sense does not have any of the problems of the aforementioned alternatives. Ground height can be represented in a continuous manner and representing terrain below other type of structures is also possible. There are some problems with the MLS in its original formulation. For one, the use case is optimized for environments where the structures are aligned with the main axis. Floors are considered flat and parallel to the xy-plane and vertical structures are aligned with the z-axis. This is a useful assumption in urban scenarios, but does not hold in general, especially not for unstructured outdoor environments. Further there are a number of parameters that govern the update rules for the surface patches of the MLS, which need to be changed depending on the scenario.

A visual comparison of some of the different forms of environment representation discussed so far is given in Figure 2.1.

2.1.2 Extended MLS

An extension to the original MLS is formulated in this work, which improves the suitability for the use in unstructured outdoor environments. For a detailed description of the original MLS formulation, the reader is referred to the work from Triebel et al. [2006]. In this section, a mathematical formulation of the environment model is given, as well as a set of rules for updating the structure with new measurements. Significant differences to the original MLS formulation are highlighted and explained. In general, the MLS is a grid structure $G = \{g_{ij}\}$

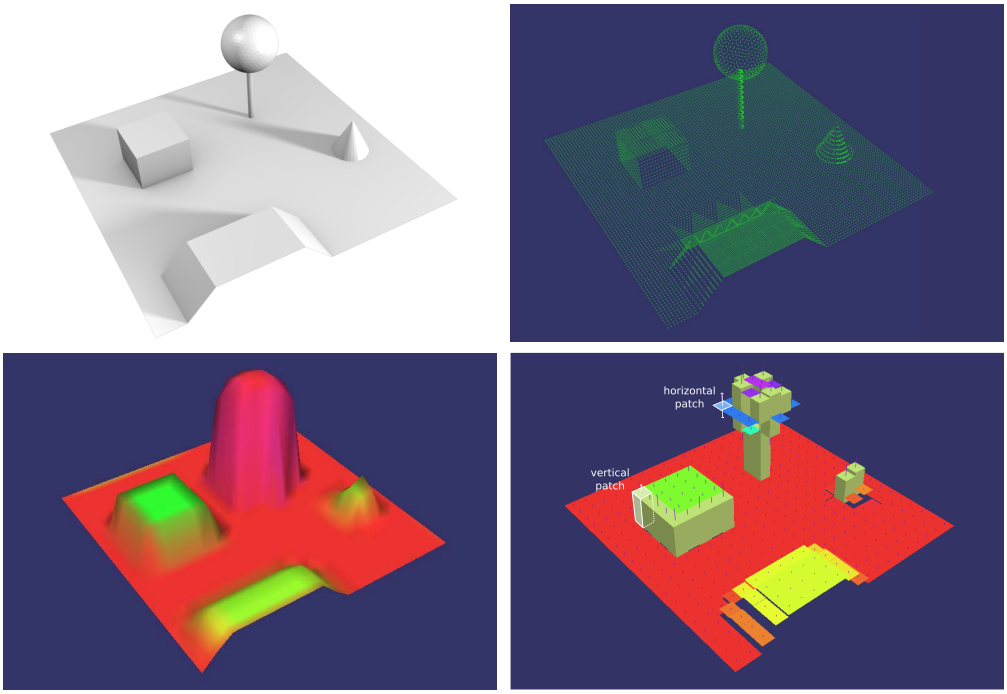


Figure 2.1: Illustration of an artificial environment scene (top left) and different ways for representing the information as an environment model. The pointcloud model (top right) requires a lot of memory and has costly lookup times. Digital Elevation Map (DEM) models (bottom left) are cheap but have drawbacks in what they can encode. The Multi Level Surface Map (MLS) representation (bottom right) provides a trade-off between the different criteria.

with ordered grid elements g_{ij}, where $i = 1..m$ and $j = 1..n$. The map has a cell size $s = (s_x, s_y)$ with $s_x, s_y \in \mathbb{R}^+$ which associates each grid cell g_{ij} with a Cartesian coordinate $c_{ij} = s \cdot (i + 0.5, j + 0.5)$. Here, c_{ij} is the centre position of the respective cell on the xy plane. Each grid cell $g_{ij} \subset S$ contains 0 or more surface patches from the set S.

The original MLS formulation is that $S = \mathbb{R} \times \mathbb{R}^+ \times \mathbb{R}_0^+$ so that a patch $s \in S = (z, \sigma, h)$ specifies the mean vertical position of the patch z, the standard deviation of this position value σ, and a height interval value h. A height value of $h = 0$ implicitly marks a patch as horizontal. An initial set of patches for g_{ij} is generated from measurement points that are within the volume of the cell. Points are processed in an ordered manner and two thresholds are used. The set of points are classified into intervals such that minimum distance between any

points of two intervals are at least t_{gapsize} apart. If the maximum distance between any points within an interval is less than $t_{\text{thickness}}$ the interval is considered a horizontal, otherwise a vertical patch. To allow vertical patches, $t_{\text{thickness}}$ must always be smaller than t_{gapsize}. A horizontal patch stores z and σ of all points in the interval. A vertical patch only stores the z and σ of the highest point in the interval, as well as the distance h to the lowest point. Whenever a point is added to the cell, it is merged with an existing cell in case it is within t_{gapsize} of an existing interval. If it is also within $t_{\text{thickness}}$, the patch stays horizontal, otherwise it becomes a vertical patch. The merging is such that for a vertical cell, the z and h are adjusted such that the new point is also included in the interval. For horizontal patches the Kalman update rule (see Section 1.2.2) is used to update z and σ.

This formulation works well enough and was used for most of the experiments in this work. It was however noticed that this is not an optimal formulation for outdoor environments. The problem is with the interpretation and update of the horizontal patches. These patches are modelled to be completely aligned with the xy-plane. The σ value gives the uncertainty over the position of the plane. Any new measurement is considered on the plane. For outdoor terrains where most of the terrain is not completely horizontal, this model assumption does not longer hold. In such cases a representation of the height distribution of the patch would actually be more helpful. This is especially the case when evaluating contact likelihoods, as it improves the estimation. The vertical patches of the MLS are already interpreted in this way. They represent a uniform distribution over the height interval. For this reason a new representation is proposed in this thesis, which does not make the distinction between horizontal or vertical patches, but models a statistic over the point distribution in the surface patch interval. The principle difference between the MLS and the eMLS is shown in Figure 2.2.

In order to model the point distribution within a given interval, the weighted mean and standard deviation formulation are used. The reason for using the weighted formulation is that each point which is added to the interval comes with a height value z_i and a measurement uncertainty σ_i. It can be shown that an unbiased estimator of the mean of the point distribution for such measurements

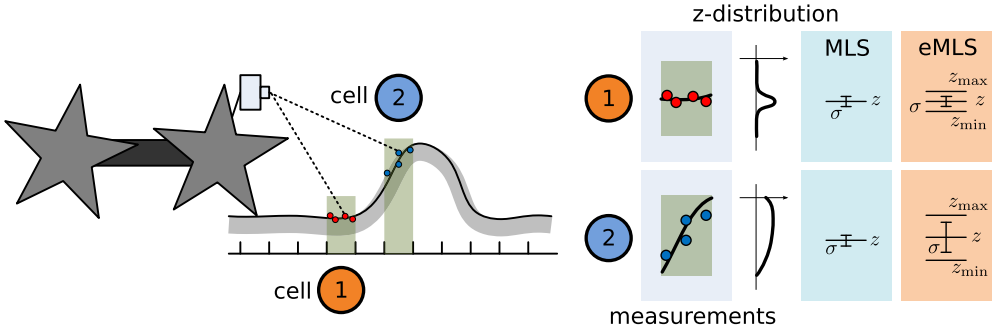


Figure 2.2: This figure shows how the standard MLS and the extended MLS (eMLS) treat different terrain measurements. The measurements in the horizontal patch (1) are treated similarly for both the MLS and the eMLS case, since the distribution of the z-values in the patch is very narrow. On the sloped patch (2), the distribution of z-values is spread further apart. While the MLS will estimate the same error distribution as for the horizontal cell, the eMLS provides better statistics on the actual z-distribution within the cell.

is

$$\hat{\mu} = \frac{\sum_{i=1..n} w_i z_i}{\sum_{i=1..n} w_i} \tag{2.1}$$

where $w_i = \frac{1}{\sigma_i^2 + \sigma^2}$. σ^2 is the variance of the point distribution. Note here that the weight in principle consists of two different variances. The uncertainty over the measurement, as well as the variance of the point distribution. An estimator of this variance is given by

$$\hat{\sigma^2} = \frac{\sum_{i=1..n} w_i}{(\sum_{i=1..n} w_i)^2 \sum_{i=1..n} w_i^2} \sum_{i=1..n} w_i \left(z_i - \hat{\mu}\right)^2, \tag{2.2}$$

which can easily be transformed into a formulation which can be used in a cumulative fashion.

$$\hat{\sigma^2} = \frac{\sum_{i=1..n} w_i}{(\sum_{i=1..n} w_i)^2 \sum_{i=1..n} w_i^2} \left(\sum_{i=1..n} w_i z_i^2 - \frac{\sum_{i=1..n} w_i z_i}{\sum_{i=1..n} w_i} \right) \tag{2.3}$$

Taking these interpretations into account a new formulation of s can be given:

$$s \in S = (\Sigma z, \Sigma z^2, \Sigma w, \Sigma w^2, z_{\min}, z_{\max}) \tag{2.4}$$

Here $z_{\min}$ and $z_{\max}$ hold the minimum and maximum z-values in the patch. Σz and Σz^2 are the sums of all $w_i z_i$ and $w_i z_i^2$ values in the patch, and Σw and Σw^2 the sum of all w_i and w_i^2 values. The $\hat{\mu}$ and $\hat{\sigma^2}$ value – so the mean and variance of the matter distribution – for each patch can then be calculated using (2.1) and (2.3). The $z_{\min}$ and $z_{\max}$ values are used to provide proper boundaries for the patch. Similar to the original MLS, a patch is merged with another patch, when the gap between the patches is smaller than t_{gapsize}. Effectively the environment model formulated this way provides a statistic over the point measurements and has benefits over the MLS. The ability to hold a statistic over the distribution of matter in the patch is one of the design criteria. Another benefit is that the sum of weights Σw is effectively a measure of the information contained in a cell. The higher this value the better the estimate of the distribution. This can be used to detect outliers in the sampling, and for example only use cells with an information higher than a threshold value, or to weight measurement models based on the information content.

In addition to the definition given in (2.4), additional meta-data can be stored with the patch. What meta-data is stored depends on the application and measurement models selected. The measurement models which are discussed in this work use combinations of the following additional datums: terrain classification vector, terrain slope and an index which holds information on which measurement had the highest update influence on the patch. This meta-data is not mathematically defined here, but can be thought of as associated with s.

The access model which is used for the application of the map structure is such that for a given point $p \in \mathbb{R}^3$ in the local frame of the map, and an interval $l \in \mathbb{R}$, the map function

$$m(p,l) = \begin{cases} s \in S & \text{surface with } z \in [p_z - l/2, p_z + l/2] \\ \emptyset & \text{no surface in interval} \end{cases} \tag{2.5}$$

provides a surface patch value s from the set of surface patches S given the point p and the search interval l. The x and y components of p specify the grid cell. Since each grid cell contains a set of surface patches at different elevations, the z component of p, p_z together with the search interval l are used to identify the

relevant patch. The patch with the largest mean elevation is used in case there are multiple patches in the interval. If no patch is within the interval, m returns the empty set.

2.2 Odometry Model

In Section 1.2.1 the general Bayes filter was introduced as a tool for estimating the probability distribution over the pose of a robot. One crucial part is the prediction step. Based on control inputs and a general notion of the system and the environment, a relative change in pose between two time steps is predicted. The name odometry stems from the Greek word *hodós – path*. Odometers are found in cars and bicycles, where they count the revolutions of the wheel. By assuming that the surface of the wheel is in contact with the ground and there is minimal slip, the travelled distance can be estimated by multiplying the revolutions with the wheel circumference. When information on the steering is included, models for the specific robot configuration provide an estimate of the full pose change. For wheeled systems for flat 2D surface, various practical solutions exist (e.g. Siegwart and Nourbakhsh [2004]; Thrun et al. [2005]), especially for low slip configurations like Ackerman-steering. With the information from the wheels alone it is possible, but difficult to estimate the 3D path of the system.

Recently, Attitude and Heading Reference Systems (AHRS) based on MEMS technology have become available at a mass and price range, where they can be incorporated into nearly every new system design. An exception to this is the availability for space certified systems (Rehrmann et al. [2011]). AHRS make use of an Inertial Measurement Unit (IMU), which includes accelerometers, solid-state gyroscopes and where applicable magnetic field sensors. The gyros measure the rotational velocity, independent of the reference system. The accelerometers sense the dynamic acceleration of the system but not the acceleration from the gravitational force. In many cases the assumption can be made that the system does not accelerate continuously into one direction, and thus that the mean can be used to estimate the acceleration which goes against the gravity vector. In this way the orientation of the system can be corrected for the pitch and roll rotations. In environments where it is feasible, a measurement of the magnetic field and

assumptions on the correlation with the earth magnetic field can be used to fix the orientation around the yaw axis as well. This assumption is in practice less stable, as the magnetic field – unlike the dynamic acceleration – can be biased for a prolonged period.

With the aid of the AHRS measurements an estimation of the path in 3D can be given. In Schwendner and Joyeux [2011] a solution is presented, which uses a simple skid-steering wheeled model and extends it to the 3D case. This model is in approximation also applicable to leg-wheel hybrid systems like the Asguard (see Section 6.1.1). Wheel models make the assumption of a continuous surface which can be in contact with the environment. This does not generally hold for legged or hybrid systems. Odometries for legged systems, which are based on discrete contact points with the environment, have seen some research in the past (e.g. Gassmann et al. [2005]; Lin et al. [2006]) and often depend on the class and configuration of the system. The work presented here uses an approach which is not systems specific and only depends on the contact points with the environments as well as an estimation of the orientation change.

The general idea is that a certain class of systems has discrete contact points with the environment. These contact points stay mostly fixed in the frame of the environment. Systems which fall into this class are leg-wheel hybrids, as well as legged systems. Within the body frame, these contact points are moving. By associating these contact points in the body frame at different time steps, an estimate on the relative movement of the body frame with respect to the environment frame can be given. This identification of body-environment contact points over different time-steps can be considered an embodied-embodied data association as defined in Section 1.1. For an intuitive interpretation of this association, the movement of a horse can be considered. When one foot is in contact with the ground, this contact point is fixed in the environment frame. This fact is illustrated very well by the foot-print of the horse-shoe that is left. By associating the position of the foot in a horse relative frame, at the time of first contact and at the time the contact is lost, one constraint is provided for the relative movement of the horse. Figure 2.3 shows an illustration of this association. The difficulties with this approach is that the system can get over-constrained when the orientation and more than one contact point is given. The model fails

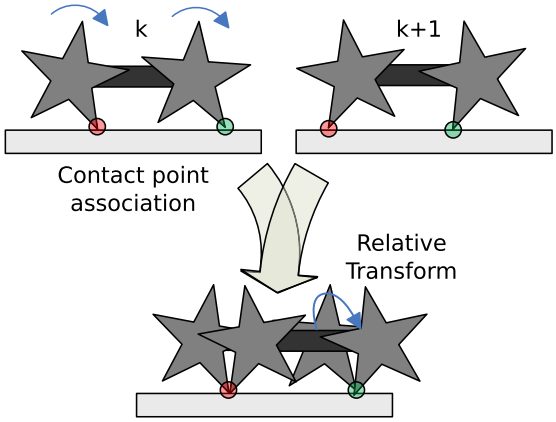


Figure 2.3: Embodied data association based odometry model. By associating the contact points from two body configurations the odometry model can provide an estimation of the relative pose transform between the two poses.

when no contact point with the environment is provided like in the case of a gallop, or when all feet slip at the same time into the same direction.

The following is a mathematical description of this contact based odometry mainly based on Schwendner and Joyeux [2011]. It also introduces some notations on frames and transformations between them, which is followed through the reminder of this work.

All the frames are defined such that the axis use a right-handed coordinate system, where the positive z-axis is up. Where applicable, the y-axis goes in the forward or north direction, and the x-axis is right or east. This convention is chosen as it seems to be most intuitive when using projections onto the xy-plane. Note that an x-forward convention is also used very often in the literature. For the actual calculations, the choice of convention makes no difference.

What we are looking for in the odometry model is the full isometry transform $C^{B_k}_{B_{k+1}}$ from the body frame B at time $k+1$ to the frame B at time k. Going from $k+1$ to k instead of the other way around, has the advantage that the actual transformation matrix is more intuitive. A positive translation in the y direction will be visible as such in the matrix, and not as its inverse.

How the transform is represented in the actual implementation is not so relevant here, and we can assume $C \in \mathbb{R}^{4\times4}$ to be a homogeneous transformation matrix with isometry constraints. As stated in the beginning of this section,

the odometry estimation can be split into a translational T and a rotational component R.

$$C_{B_{k+1}}^{B_k} = \begin{bmatrix} R_{B_{k+1}}^{B_k} & T_{B_{k+1}}^{B_k} \\ 0 & 1 \end{bmatrix} \tag{2.6}$$

Note that $C_{B_{k+1}}^{B_k}$ is the estimate given by the odometry, and not the true transform $\bar{C}_{B_{k+1}}^{B_k}$.

The rotational part $R_{B_{k+1}}^{B_k}$ can be directly derived from the estimation of the AHRS. The AHRS itself provides the rotation R_{AHRS}^{W} from its own reference frame AHRS to a geo-fixed world frame W. For the remainder of this work we consider W to be a Cartesian frame where the y-axis is north, so the curvature of the Earth or planetary surface is not taken into account. For Earth this is a valid approximation for areas multiple hundred kilometres across.

Since the AHRS unit is in nearly all cases rigidly connected to the body of the system, there is a transform C_{AHRS}^{B}, which is either known from the construction of the system, can be tape-measured or calibrated using actual measurements and a ground reference. Since we are only interested in the rotational part R_{AHRS}^{B}, knowing the orientation of the device is sufficient for application in the odometry. With this information, the orientation for the odometry (2.6) can be given as:

$$R_{B_{k+1}}^{B_k} = (R_{\text{AHRS}_k}^{W} R_{B}^{\text{AHRS}})^{-1} R_{\text{AHRS}_{k+1}}^{W} R_{B}^{\text{AHRS}} \tag{2.7}$$

For associating body-environment interactions for different time step, the assumption is made that these interactions happen at discrete contact points. For technical reasons we define the number of these potential contact points as $N_{\mathfrak{C}}$, and the points as

$$\mathfrak{C}^{\text{can}} = \{c_1, c_2, \ldots, c_{N_{\mathfrak{C}}}\} \tag{2.8}$$

with $c_i \in \mathbb{R}^3$ given in the body coordinate frame B. For most systems these candidate points are the nominal points, where the system can be in contact with the environment. So, for example, on a legged system this would include the feet but exclude the camera (we hope). Of course the position in the body frame of these contact points can change between different time steps, and actually needs to change to get the system moving.

Also, not all of these candidate points are in contact with the environment for a given time step k. The information if a candidate point is active, and thus in contact with the environment is given by:

$$\mathcal{C}^{\text{act}} = \{a_1, a_2, \ldots, a_{N_\mathcal{C}}\} \tag{2.9}$$

The individual elements $a_i \in \{0, 1\}$ of this activation vector denote if the contact point c_i is either inactive 0 or active 1. The information if a candidate point is active or not can be provided by sensors (see Section 6.1.2), or in some situations estimated as will be shown later in Section 2.3.1.

Based on (2.8) and (2.9) we can let $\mathcal{C}_k$ be the set of active contact points for a time step k.

$$\mathcal{C}_k = \{c_i^k \in \mathcal{C}_k^{\text{can}} | a_i^k \in \mathcal{C}_k^{\text{act}}, a_i^k = 1\} \tag{2.10}$$

In order to find the distance the system has moved between two time steps – the translational component of (2.6) – we first have to find the contact points which have stayed fixed in the world frame W between these two time steps. Because of slip effects and other problems this can only be achieved approximately by finding those points which have been active in both consecutive time steps. For this purpose we define an association function which provides point pairs where this is the case.

$$\mathcal{A}_{k,k+1} = \left\{(c_i^k, c_i^{k+1}) | a_i^k, a_i^{k+1}, a_i^k = a_i^{k+1} = 1\right\} \tag{2.11}$$

Note that this is similar to (2.10) but the set associations are left out for readability.

With this set of associated points we can approximate the translation of the system by rotating the second contact point in the pair into the same coordinate frame as the first point. An estimate of this rotation is of course known through (2.7). The translation for the odometry should now be such that the two points are on the same position in the shared reference frame. If there is only one contact pair, so $|\mathcal{A}_{k,k+1}| = 1$, there is a unique solution. With any more pairs the problem is over constrained. One possible solution would be to minimize the squared error. It can be shown that this is achieved by taking the mean of all translations. For a homogeneous surface and an equal weight distribution over the contact points

this is a good approximation. In practice, variability of terrain properties and the weight distribution of the system needs to be taken into account, though. So instead of the mean, a weighted mean is used to finally calculate the translational component.

$$T_{B_{k+1}}^{B_k} = \frac{1}{\sum w_i} \sum_{(c^k, c^{k+1}) \in \mathcal{A}_{k,k+1}} w_i (c^k - R_{B_{k+1}}^{B_k} c^{k+1}) \tag{2.12}$$

with w_i the weight associated with the contact point c_i^k. (2.12) of course just reverts to the normal average if $w_i = \frac{1}{|\mathcal{A}_{k,k+1}|}$.

The combination of (2.7) and (2.12) form the full transform (2.6) of the robot body between k and $k+1$. The model that has been described only covers the main effects of the transformation under certain assumptions. There are likely many other influences which are not included in the model, or which have an error in the modelling assumptions. As discussed in Section 1 we use Bayesian probability to model these errors in a stochastic way. More specifically we want the odometry to be the forward prediction part in the Bayesian filter equation (1.11). In practice, the odometry formulation is used for each time step where the AHRS provides new orientation values. The iterative convolutions of the Bayes filter equations will let the distributions approach a Gaussian distribution as described by the central limit theorem. For this reason it makes little sense to use anything but a Gaussian distribution for errors that are uncorrelated between step k and $k+1$. Errors that are correlated like for example the weights w_i^k for the mixing of the translations actually need to be included in the state. In this thesis only the uncorrelated errors are considered.

In that case we can formulate the state of the system as

$$x_k = (x, y, z, \phi, \psi, \theta) \tag{2.13}$$

which is basically the full geometric transform $C_{B_k}^W$ from the body frame B at time k to the world frame W. x, y and z are the translational components and ϕ, ψ and θ are the roll, pitch and yaw angles of the rotation respectively. The actual

transform can thus be given as a function of the state

$$C(x_k) = C_{B_k}^W \tag{2.14}$$

so that even if we later redefine x_k to contain additional state information we can still express the transform associated with it.

The Bayes prediction formula requires the probability distribution for the next state x_{k+1} given the previous state x_k. The odometry model developed in this section is entirely relative to the previous state, so that the prediction formula can be given as:

$$p(x_{k+1}|x_k, u_k) = \mathcal{N}(C(x_k)\mu_O, C(x_k)\Sigma_O C(x_k)^T) \tag{2.15}$$

where μ_O is the mean and Σ_O the covariance of the odometry, and u_k is the odometry input consisting of the contact point values and the orientation reading from the AHRS. The mean in our context is estimated by $\hat{\mu_O} = C_{B_{k+1}}^{B_k}$, while the covariance matrix is estimated based on different influence factors given by:

$$\hat{\Sigma_O} = diag\left(A\begin{bmatrix}\text{tilt}\\ \Delta\text{translation}\\ \Delta\text{heading}\\ 1\end{bmatrix}\right) = diag\left(A\begin{bmatrix}\cos^{-1}(e_z \cdot R_{B_k}^W e_z)\\ |T_{B_{k+1}}^{B_k}|\\ R_z^{-1}(R_{B_{k+1}}^{B_k})\\ 1\end{bmatrix}\right) \tag{2.16}$$

Where e_z is the unity vector in the direction of the z-axis. $R_z^{-1}(C)$ is the rotation around the z axis of the transform C. $A \in \mathbb{R}^{6\times 4}$ is a configuration matrix, which relates the individual factors to the error covariance matrix in a linear way.

The individual error influences have different origins:

Constant error is mainly useful for the orientation error, in order to model gyro drift on the heading. Another possibility to compensate the heading drift is to assume zero velocity updates. When all contact points have stayed fixed within a certain error interval for a certain period, it is possible to assume that the system is not moving. In this case the rotation from the AHRS can be overridden by the identity rotation.

Tilt is the angle of the system against the gravity vector. When the system is on an inclined terrain, it is more likely to exhibit slip and other dynamic effects, as the forces parallel to the plane of movement are much higher compared to when the terrain is flat.

Delta heading is the change in heading since the last update, so effectively the rotation speed. Especially on skid steering systems, the rotation has a large influence on the rotation error, since the point of rotation is not identifiable precisely and depends on the terrain shape and properties.

Delta translation is the translational velocity of the system. With higher velocity, the influence of dynamic effects will increase. The odometry model as it is, does not really take dynamic effects into account, they are however factored through delta heading and delta translation.

The actual values for the coefficients of A depends on the system and the environment, and have to be determined empirically.

The model described so far works under the assumption that the body-environment contact information $\mathcal{C}_k^{\text{act}}$ is actually known. This requires sensors to determine the contact information. On systems with many more contact candidates than actual contact points like for example the Asguard, it is not always practical to have the contact points measured by sensors. Sensors, especially in the feet are error prone and often fail due to material wear.

On systems like the Asguard, an additional assumption can be used to estimate the contact information. Because of the geometry, and assuming a quasi-static model, for each of the wheels there is always at least one contact point per wheel. To generalise this, we could generate a partition of $G = 1 \dots N_{\mathcal{C}}$, which is the set of indices each representing a candidate contact point. There are N_G non-empty groups in this partition, so that $G_i \subset G$ for $i = 1 \dots N_G$. A trivial estimator of the contact vector would be to take the point with the lowest z-position for each group. Using (2.8) and (2.9) we can define $\mathcal{C}_k^{\text{act}}$ over its coefficients:

$$a_i = \begin{cases} 1 & i \in G_j, \forall k \in G_j : (c_i)_z \leq (c_k)_z \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

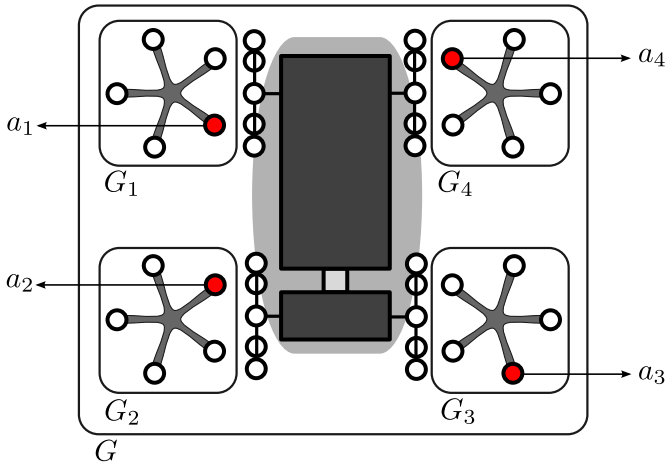


Figure 2.4: In case the activity a_i of contact points can not be sensed directly, the entire contact point group G is partitioned into subgroups G_i. An estimate of the contact activity is given by choosing the point with the lowest z-value within each group.

for $i = 1 \dots N_{\mathcal{C}}$. An illustration of this concept is given in Figure 2.4

2.3 Contact Model

The body-environment contact based measurement model is the first of three visual-embodied measurement models which are described in this work. Measurement models are used in the Bayes update step (1.12) and give the likelihood distribution of a measurement given the state. The principle idea behind the contact model is to evaluate the likelihood of the system configuration and its contact points with the environment for a certain position in the map. This is illustrated in Figure 2.5. The position of the contact points in the body frame B is known from the forward kinematic model and joint encoder angles. Also, the orientation with respect to the gravity vector is given by the AHRS. What is unknown is the position in the map and in the 3D case the heading. The aim of the measurement models is now to evaluate the likelihood of getting a measurement for the contact points (effectively from the joint encoders and the AHRS) *given* a position and a map.

All the measurement models which will be described in this thesis are formulated to fit (1.12), and also include the map m as conditional in addition to the

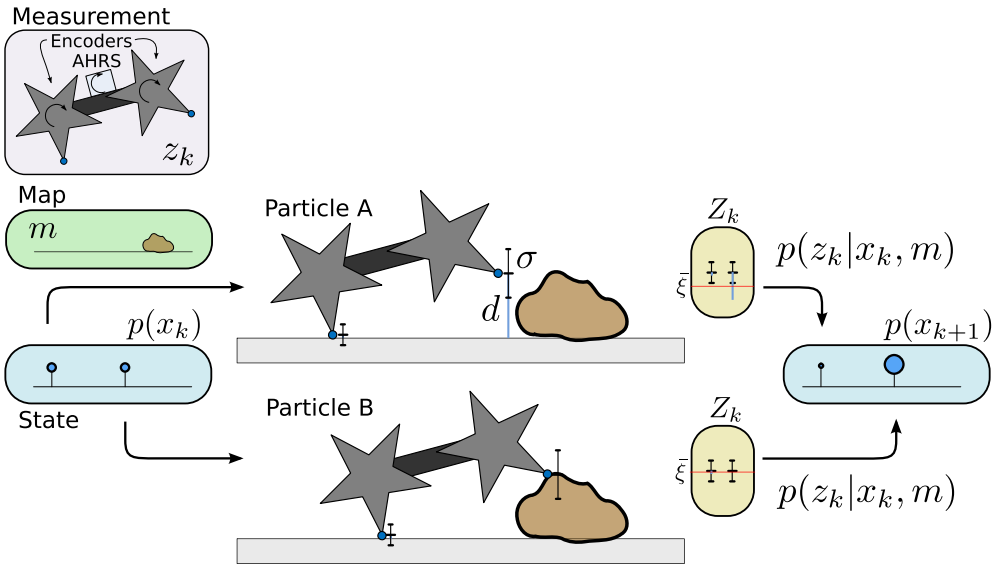


Figure 2.5: The aim of the body-contact measurement model is to evaluate the likelihood of a measurement z_k given a map m and a state x_k. The measurement consists of the body-environment contact points and parts of the orientation coming from the AHRS. The illustration shows how the measurement likelihood is evaluated for two different states and how these evaluations influence the probability density function of the state in a Bayes filter set-up.

state x_k.

$$p(z_k|x_k, m) \tag{2.18}$$

x_k and m will be defined the same way for all the models. The variation of the measurement models is how z_k is defined. For the body-environment contact model, the measurement $z_k = (\mathcal{C}_k, R_B^{Y_k})$ is defined as the set of contact points $\mathcal{C}_k$ at time k as given in (2.10), as well as the rotation from the body frame B to the yaw compensated orientation frame Y at time k. Figure 2.6 shows amongst other things the relation between Y and B. The idea of the Y frame is that it does not take any rotation around the yaw axis into account. To form the full rotation $R_{B_k}^W$, the yaw rotation is taken from the state $C(x_k)$. This leads to the mixed notation for the full transform as a combination of x_k and z_k.

$$C_{B_k}^W = \begin{bmatrix} R_z(C_R(x_k))R_B^{Y_k} & C_T(x_k) \\ 0 & 1 \end{bmatrix} \tag{2.19}$$

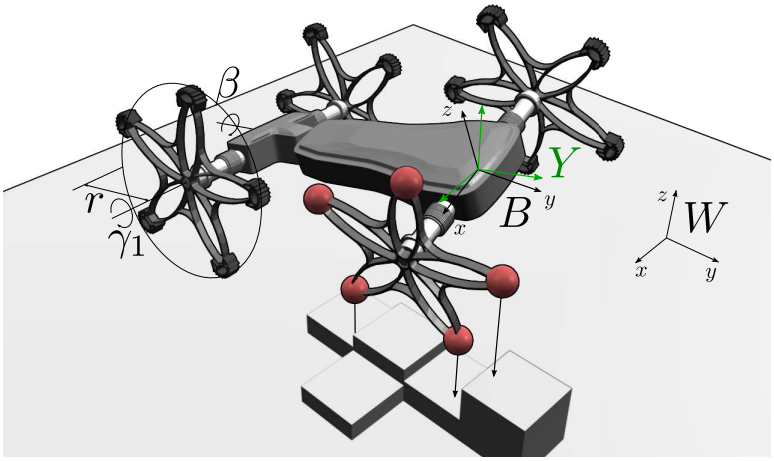


Figure 2.6: In order to estimate how well a given set of contact points given in frame B fit an environment, they are transformed into coordinates in the world frame W, and the contact points (red) compared to the height estimates from the map. The Y frame is a rotation frame where the xy-plane is parallel to the xy-plane of W, but the projection of the-y axis is equal to the-y axis of B.

The subscripts C_T and C_R indicate the translational and rotational components of the full transform. The reason for splitting the transform into measurement and state is that, later we do not have to model the uncertainty over the pitch and roll axis of the state explicitly, but can consider it part of the measurement.

In order to test the likelihood for a combination of $\mathcal{C}$ and $C^W_{B_k}$ given a map m, the foot positions in frame W are compared to the map. This is done by essentially treating each contact as a single measurement and combining the individual likelihoods into a joint likelihood.

Given the full transform and the contact points in body coordinates, we can now evaluate the distance of the contact points to the map. The distance is only evaluated in the z-axis, for reasons that have to do with the representation of uncertainty for the state. This is discussed in greater detail in Section 3.1. For a single contact point $c \in \mathcal{C}$ the distance to the map value d_z as well as the

uncertainty of this distance σ_d is evaluated as

$$
\begin{aligned}
(m_z, \sigma_{\text{map}}, \cdot) &= m(C_{B_k}^W c, l) && (2.20)\\
d_z &= m_z - (C_{B_k}^W c)_z && (2.21)\\
\sigma_d &= \sqrt{\sigma_{\text{map}}^2 + \sigma_{\text{meas}}^2}, && (2.22)
\end{aligned}
$$

where m is the map function (2.5) with the search interval parameter l, which is set to be three times the position uncertainty on the z-axis σ_z. σ_{meas} is the measurement uncertainty, which depends on the system and the environment. This error includes all effects which are not explicitly addressed in the model, like for example a sinking of the contact point in soft ground.

Equation (2.21) and (2.22) give ground distance values and associated uncertainties for a single point. These pairs can be combined into a set of distances and their uncertainties for all contact points for which there is map information available. Remember that the map function m is defined to return the empty set if no information is available at a particular position.

$$Z_k = \{(d_z, \sigma_d) | c \in \mathcal{C}_k, m(C_{B_k}^W c, l) \neq \emptyset\} \tag{2.23}$$

This set of distances Z_k effectively determines the likelihood of a measurement z_k given the map and the state. The idea is to take the joint likelihood of the individual distances. For this we normalize the distances with their respective σ value, and take the product of those values evaluated through the normal distribution ϕ.

$$\theta(Z_k, \xi) = \prod_{(d,\sigma)\in Z_k} \phi(\frac{d+\xi}{\sigma}). \tag{2.24}$$

The offset value ξ is used to shift the measurements along the z-axis. For the evaluation of the likelihood of a full state, which already includes a z-position, ξ is 0. Based on this likelihood value for the set Z_k, the measurement probability can be given as a function of θ, which includes z_k, x_k and m through (2.19)-(2.23).

$$\hat{p}(z_k | x_k, m) \propto \theta(Z_k, 0) \tag{2.25}$$

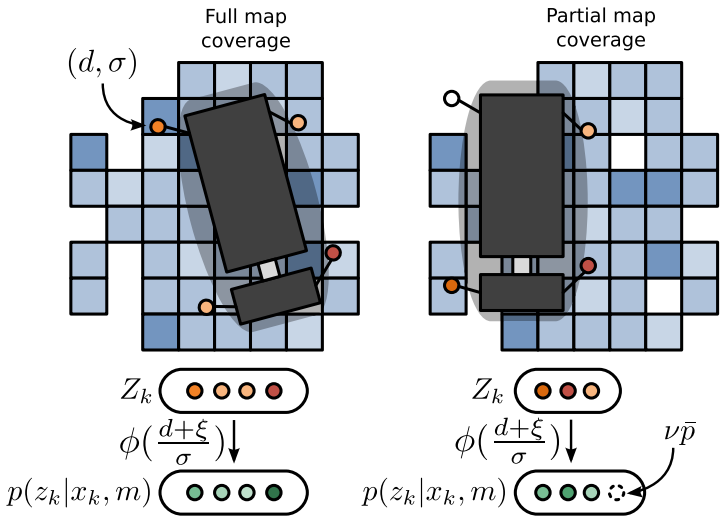


Figure 2.7: For a state x_k, each active contact point with the environment has a distance and uncertainty pair, if map information is available at that point. The set of these pairs Z_k is used to evaluate how well that particular state fits the environment. In order to make these likelihood values comparable, discounted average contact likelihoods ($\nu\bar{p}$) are used to fill up sets where only some of the contact points have map values.

This is not yet normalized for the set size $|Z_k|$. The problem is that for different instances of x_k, the size of Z_k varies. At different locations in the map their may not be information available. With variable sizes of the set, (2.24) will be biased towards sets with a low number of distance values. The question is how to handle this situation. We choose to use a mean value for the distance likelihood and add it to the product. Let's say the largest number of distances in a set is N, we effectively fill up the slots up to N with a discounted mean probability value $\bar{p}$. The discount value $\nu < 1$ is used so that we favour known map cells over unknown ones. Figure 2.7 gives an illustration of this concept. With this we can finally provide the measurement probability as

$$p(z_k|x_k, m) \propto \theta(Z_k, 0)(\nu\bar{p})^{N-|Z_k|} \tag{2.26}$$

The mean probability $\bar{p}$ is the mean of all contact likelihoods. Both $\bar{p}$ and the maximum amount of elements in Z_k depend on how the measurement model is used, and further descriptions are given in Section 3.1 on the localisation filter.

What has been developed so far is the likelihood of a given measurement for a pose and a map. Another interesting aspect the body-environment contact point model can provide is the z-position for which this measurement likelihood is at a maximum. This effectively provides a derived measurement, which can be used to correct the estimation of the z-component of the system pose.

In order to do this, we seek the ξ for which (2.24) is at a maximum. By taking the derivative of the log-likelihood and setting it to zero, we get the maximum value at

$$\bar{\xi} = \arg\max_{\xi} \theta(Z_k, \xi) = \left(\sum_{(d,\sigma)\in Z_k} \frac{d}{\sigma^2} \right) \left(\sum_{(d,\sigma)\in Z_k} \frac{1}{\sigma^2} \right)^{-1}, \tag{2.27}$$

which is effectively the weighted mean for a heteroskedastic measurement. Each individual distance measurement is normalized by its variance.

To use this information as a measurement we would also like to get a measurement uncertainty. It turns out this is given by

$$\sigma_{\xi}^2 = \left(\sum_{(d,\sigma)\in Z_k} \frac{1}{\sigma^2} \right)^{-1}. \tag{2.28}$$

2.3.1 Contact Point Activity Estimation

As with the odometry model, the contact model from Section 2.3 makes the assumption that the body-environment candidate contact points are known, and which ones are active. Remember that candidate contact points are those that could be in contact with the environment, but actual contact is indicated by an active flag. If there are no sensors available to measure if a contact point is active, the contact points activity have to be estimated. Unlike the activity estimation in the odometry case, the contact model has an additional source of information. The distance values d for each contact point, which is the distance in the z-axis between the candidate point for a specific state x_k and the map. The reasoning is that candidate points with low distance values have a higher change of actually being in contact compared to those with higher distance values. So the idea behind the contact activity estimation for the contact model is to evaluate the

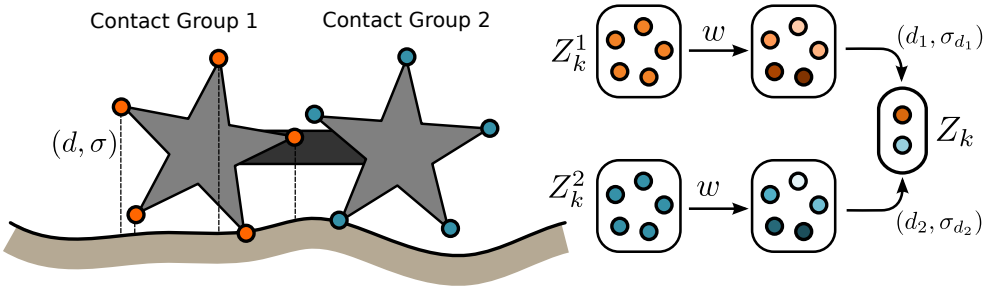


Figure 2.8: When no contact activation information is available, one possibility of estimating the activation information is by using the (d, σ) values for each candidate contact point and to combine them in group sets. The individual values are weighted according to the function w and then aggregated into individual measurement values, resulting in a measurement set Z_k, which can be used in the contact estimation model.

distance values for all candidate contact points within a group, and then mix these values into a single distance and distance uncertainty measurement for the entire group. The contact groups were already introduced in the odometry section and represent a subset of all candidate points, where at least one of the candidates is active, but it is not known which one. The aggregated measurement values for the group can then be used as normal measurements in the contact model. See Figure 2.8 for an illustration of the concept.

So instead of using the points in the contact group $\mathcal{C}$ in (2.23) we define N_G measurement groups using the same partition method for the candidates as in the odometry case.

$$Z_k^i = \{(d_z, \sigma_d) | j \in G_i, c_j \in \mathcal{C}_k^{\text{can}}, m(C_{B_k}^W c_j, l) \neq \emptyset\} \tag{2.29}$$

for $i = 1 \dots N_G$. Each contact group has its own set of distance values and uncertainties now. To join these values into a single distance and uncertainty value per group, a linear average between the member of a group is generated, so

that

$$\eta = \sum_{(d,\sigma_d)\in Z_k^i} w(d,\sigma_d) \tag{2.30}$$

$$d_i = \frac{1}{\eta} \sum_{(d,\sigma_d)\in Z_k^i} w(d,\sigma_d) d \tag{2.31}$$

$$\sigma_{d_i}^2 = \frac{1}{\eta} \sum_{(d,\sigma_d)\in Z_k^i} w(d,\sigma_d) \sigma_d^2 \tag{2.32}$$

where w is a weighting factor based on distance and uncertainty value, which will be explained shortly. These mixed measurements are now treated as if they where single activated measurements in the measurement set, and thus replacing (2.23).

$$Z_k = \{(d_i, \sigma_{d_i}) | i = 1 \dots N_G\} \tag{2.33}$$

The rest of the contact model stays the same.

The question that remains is how to specify the weighting function $w(d, \sigma_d)$. Intuitively this is clear, the candidate point with the lowest distance value d is the most likely point which is active. However, we have to take the measurement and model uncertainty into account, which is aggregated in σ_d. Since we have a probabilistic model, also candidates with a large distance d have a (low) probability of actually being the active contact point in the group. It turns out, the relative likelihood if a candidate point is active can be approximated by

$$w(d,\sigma_d) = \frac{\phi(\frac{d}{\zeta\sigma_d})}{\Phi(\frac{d}{\zeta\sigma_d})} \tag{2.34}$$

where ϕ is the normal distribution, and Φ the cumulative distribution of the normal function. ζ is a normalization factor, which depends on the geometry of the system. The idea behind this weighting function, which includes what is called the Mill-ratio, is that for each d it represents the ratio of active divided by non-active points, normalized for the uncertainty.

It is helpful to get an interpretation of this process. Figure 2.9 shows the distribution of contact vs non-contact cases for a simulated scenario with added

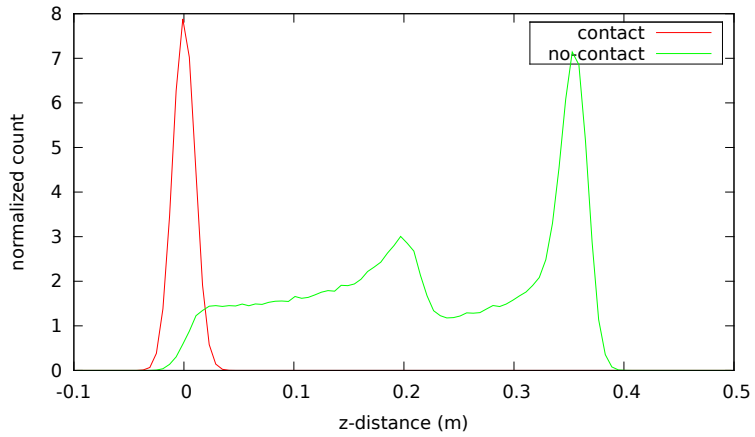


Figure 2.9: Normalized relative occurrences for estimated distance to ground in contact and non-contact cases for the Asguard system. The estimate of distance to ground is perturbed by a Gaussian noise. The shape of the non-contact distribution comes from the geometry of the Asguard wheel.

noise. The contact ratio function (2.34) estimates the ratio between contact and non-contact occurrences for a given distance value. The comparison of the estimate with the value for the Asguard case in Figure 2.10 shows that this is a valid assumption.

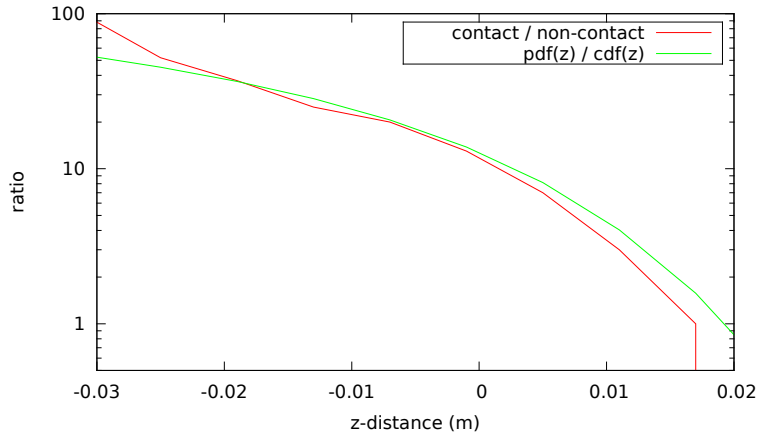


Figure 2.10: The ratio between contact and non-contact states given the distance to the ground for all five contact points of an Asguard wheel. The red curve shows the ratio for the results shown in Figure 2.9. The green curve is the estimation based on the ratio between the Gaussian density function and the cumulative density function with a suitable ζ.

2.4 Slope Model

The contact model described in Section 2.3 works by matching the shape of the contact points to the shape of the local terrain. This is likely to be effective in the presence of strong features like e.g. small boulders. Effectively that measurement model responds well to high frequency parts of the terrain shape. In case the high frequency properties are very sparse, or that the cell size of the available model is higher than the size of the system, it may be advantageous to use an alternative model (Schwendner and Hidalgo [2012]), which responds to the lower frequency parts of the terrain properties (see Figure 2.11 for an illustration).

The contact model from Section 2.3 evaluates how well the contact points fit the shape of the terrain. This implicitly includes the local slope. Three of the contact points in $\mathcal{C}$ can be interpreted as being on a plane, which provides the slope at the given position. The problem with this interpretation is that when the cell size of the environment map is larger than the system, this local slope may not provide an adequate representation of the slope averaged at the cell size. To give an example for this, one could think of a mountain which has a serpentine path leading to the top. The instantaneous slope when travelling on the path is –

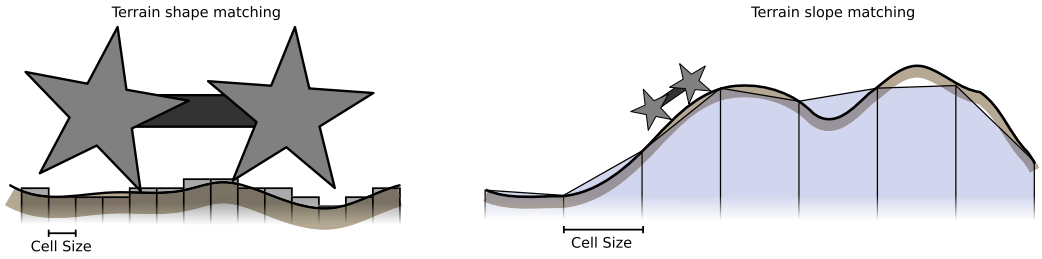


Figure 2.11: The contact model described in Section 2.3 provides a match of the local terrain shape. It is not so well suited when the resolution of the map is lower than the size of the system. In this case the slope of the terrain can be used for matching the system to the terrain.

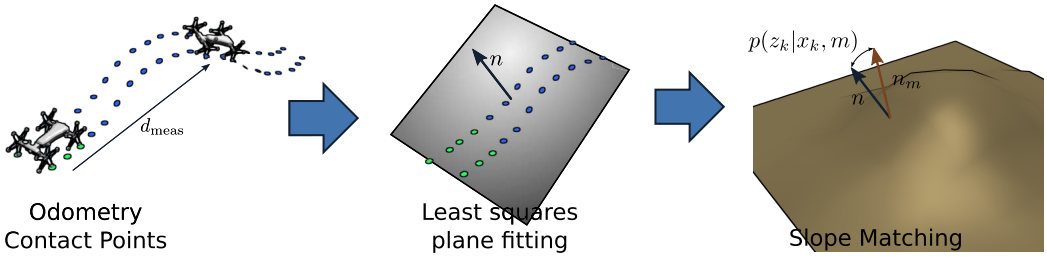


Figure 2.12: Overview of slope matching process. First the contact points are accumulated in the odometry reference frame. After a threshold distance from the origin, a plane is fitted using a least-squares method over the points. The resulting slope is then compared to the terrain slope of the model.

by design – very different from the slope of the mountain over the entire path. For that reason, another strategy is used here. It is assumed that the odometry model (Section 2.2) is accurate enough to give reasonable relative pose changes for the distance of a cell of the map model. So instead of using just the current contact points, all contact points for a certain distance travelled are accumulated in an arbitrary odometry frame of reference. Once this distance is covered, a plane is fitted through these points. Comparing the normal of that plane with the normal of the terrain at that point is the basis for the measurement likelihood given the pose and the map. The process is illustrated in Figure 2.12.

Let $C^O_{B_k}$ be the transform from the body frame at time k to an arbitrary world-fixed odometry frame O, such that $C^O_{B_{k+1}} = C^O_{B_k} C^{B_k}_{B_{k+1}}$, where $C^{B_k}_{B_{k+1}}$ is based on the odometry model. The set $\mathcal{S}$ of contact points in the odometry frame can

be constructed by transforming the contact points for the last n time steps.

$$\mathcal{S} = \bigcup_{i=1..n} \bigcup_{c \in \mathcal{C}_{k-i}} C^O_{B_{k-i}} c \tag{2.35}$$

In this approach n is chosen to be the greatest n so that the total translation $|T^{B_{k-n}}_{B_k}| < d_{\text{meas}}$ remains smaller than a threshold value. A sensible value for d_{meas} is the cell size of the map.

The next step is to fit a plane through the set $\mathcal{S}$ of contact points. There are multiple methods on how this can be done. For simplicity and performance reasons a linear regression is used, which minimizes the square-error along the z-axis. The following system of linear equations

$$\sum_{p \in \mathcal{S}} \begin{bmatrix} p_x^2 & p_x p_y & p_x \\ p_y p_x & p_y^2 & p_y \\ p_x & p_y & 1/|\mathcal{S}| \end{bmatrix} x = \sum_{p \in \mathcal{S}} \begin{bmatrix} p_x p_z \\ p_y p_z \\ p_z \end{bmatrix} \tag{2.36}$$

which when solved for $x \in \mathbb{R}^3$ results in a scaled normal of the fitted plane $\tilde{n} = (-x_x, -x_y, 1)$ which can be normalized to $n = \tilde{n}/|\tilde{n}|$. The surface normal is expressed in the odometry frame O. For the slope of the fitted plane to be comparable to the slope of the map cell, the normal vector is rotated into the body frame B at time k. Rather than using the full R^O_B rotation, only the component around the z-axis is applied to preserve the direction of the gravity vector. This results in the measured normal in body coordinates as

$$z_k = R_z(-R_z^{-1}(R^O_{B_k}))n. \tag{2.37}$$

R_z is a function that maps a rotation angle to a rotation transform around the z-axis, while R_z^{-1} is the inverse, which takes the a rotation and results in the angle of the rotation component around the z-axis.

The measurement model is used to estimate the likelihood $p(z_k|x_k, m)$ of a particular measurement z_k given the state $x_k = (x, y, z, \phi, \psi, \theta)$ defined in (2.13) and a map m. The measurement in this case is the surface normal in body coordinates, and the map is a model of the environment as described in Section 2.1. The measurement is actually a derived measurement based on the

odometry model and the contact points information, since the terrain slope can not be measured directly.

Similar to the measured normal from the odometry, the map normal needs to be rotated into body coordinates, in order to be comparable. The comparison is performed by calculating the angle between the two normals. The error due to measurement uncertainties and other effects is assumed to be normally distributed so that for the likelihood of getting a slope measurement z_k given the state x_k and the model m we can finally give

$$p(z_k|x_k, m) = \phi\left(\frac{1}{\zeta}\cos^{-1}\left(z_k \cdot R_z(-\theta)n_m([x \;\; y \;\; z]^T)\right)\right), \tag{2.38}$$

where ζ is the measurement noise factor, ϕ the standard normal distribution and $n_m(p)$ the surface normal at point p.

2.5 Terrain Classification

The two measurement models which have been proposed so far both use the direct sensorial information from the AHRS and the wheel encoders to match the body configuration to the shape or slope of the terrain. By the classification scheme discussed in the introduction section, they can be considered of the direct embodied data type. The shape and slope based measurement models only work if there is enough spatial variance in the environment to distinguish different places. Completely flat terrain for example is not suitable for these models.

In this section a different measurement model is proposed, which uses the terrain type instead of the terrain shape to perform data association. While the terrain shape can be sampled using range sensors, or estimated using stereo processing, the terrain type is more difficult to extract. In this work, terrain classification based on texture patch similarities is applied to perform the visual part in the visual-embodied data association. The embodied part is performed analysing the torque profile in the event of a wheel or leg slip. This information is associated with the locomotion behaviour of the system and the actual wheel slip events tightly connected to the control behaviour. For this reason, the slip based terrain classification can be considered to be an example for indirect embodied

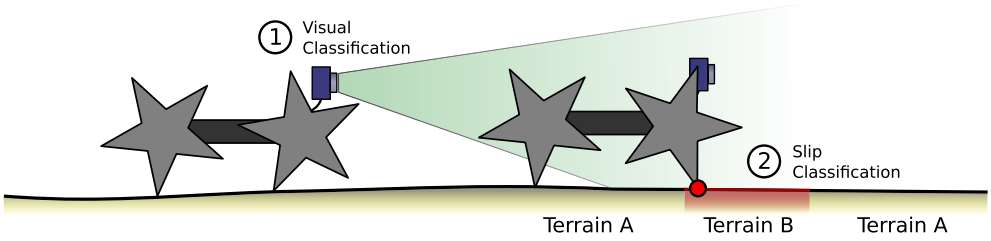


Figure 2.13: The terrain classification based measurement model maps the terrain type using a visual method (1) and correlates these measurements with slip based terrain estimation (2) later. Unlike the contact based model, this method is also suitable for terrain with little variation in shape.

data.

The terrain classification based measurement model associates the slip based terrain classification in the event of a wheel slip with the visual terrain classification at the position of the wheel slip. This visual classification for this position has been performed at some previous time step, and thus associates the current pose with the previous pose (see Figure 2.13).

2.5.1 Vision based Classification

Classifying the terrain type based on visual appearance has been extensively researched in the past, and there are numerous approaches working with color, texture patches or a combination of both (Angelova et al. [2007]; Halatci et al. [2008]). Other methods use feature descriptors like SURF or Daisy (Khan et al. [2011]) in order to achieve a scale, illumination and projection invariant vector value for comparing a sample patch to the actual test image. The work on the visual classification used in this thesis was performed by Christopher Gaudig (Schwendner et al. [2014]).

For the use of the terrain classification in the embodied SLAM context, the actual method for performing visual terrain classification is not so relevant, as long as it provides a feature vector for the classification of terrain for a subregion

of the test image.

$$T_{\text{visual}} = \begin{pmatrix} t_{\text{terrainA}} \\ t_{\text{terrainB}} \\ \dots \\ t_{\text{terrainN}} \end{pmatrix} \tag{2.39}$$

The vector T_{visual} is generated for each position in the test image, and embedded in the environment map as described in Section 2.5.3. Each of the N terrain classes $t_{\text{terrainN}} \in \{0, 1\}$ provide a binary value for a terrain match. These values are calculated using the method of histogram back-projection (Swain and Ballard [1990]).

Each of the terrains has a template image, which is an average visual representation of the terrain class. A histogram for these template images is generated in the HSV color space, which is more robust to variations in lighting conditions. Each histogram is normalised to the bin with the highest count. These histograms are stored for the processing of the test images.

Each pixel of the test image is now evaluated against all the histogram templates. The histogram bin value for the test pixel is looked up in the template histograms, and the normalised count value in the template taken as a representation for the class likelihood. The result is an image with N channels, where each pixel value in the channel represents the class likelihood. The image channels are post-processed to remove small regions and close holes. Also, the channels are binarised so that only the terrain class with the highest likelihood value is used, if it is above a threshold value. If no terrain class is rated high enough, the resulting feature vector is 0. Figure 2.14 shows an example of the input data and the resulting classifier output.

2.5.2 Slip based Classification

The interaction of a ground based vehicle with the environment has been used for performing terrain classification in numerous instances and for different locomotion modalities. The actuator torque profiles and ground contact forces in walking machines have for example successfully been used for classification by Spenneberg

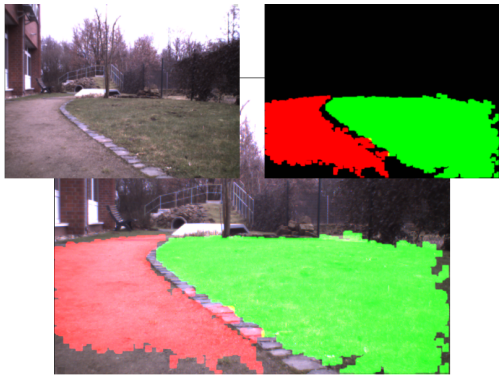


Figure 2.14: The visual terrain classification work was done by Christopher Gaudig (see Schwendner et al. [2014]), and uses histogram back-projection of template images in order to estimate the terrain classes of an output image. The input image (upper left) was classified (upper right) for two terrain classes: grass (green) and sand path (red). The overlay of input and classification image is shown at the bottom.

and Kirchner [2005] and by Hoepflinger et al. [2010]. These are prime examples for using embodied data, since the torque profiles are not just sensors, but are in the loop with the systems locomotion control system. Individual locomotion behaviours and reflexes are mixed to achieve the system locomotion. The data which is used for classification is thus not just a measurement of some environment parameter, but a measurement of the activity of the sensor-motor part of the system. A similar argumentation can be used for the wheeled systems that have been used for terrain classification (Weiss et al. [2007]; Wurm et al. [2009]) or terrain parameter estimation (Iagnemma et al. [2004]), albeit not with such a strong embodied influence.

Again, like in the visual classification case, the method for determining the terrain is not so much the focus of this thesis. The actual work on the slip based terrain classification has been performed by Patrick Paranhos (Schwendner et al. [2014]). The principle idea of the method is that terrains will have different friction coefficients and thus a variation in the traction force exerted on the body-environment contact point which is slipping. Two prerequisites are required for this method. One is the detection of slip events, so when the body-environment contact points are active, but not static with respect to the environment reference frame. The second input is the traction force, which is the force component in

the direction of the tangent of the body-environment contact point.

The method should be in principle valid for most ground based system, but has been implemented in this work only for the Asguard system. The slip detection can be estimated either based on a wheel model (Schwendner et al. [2014]), or using the odometry model from Section 2.2. The latter method is based on the influence of the individual contact point errors on the translation mean (2.12). Contact points with a large deviation from the mean are more likely to have slipped than others. So to detect single slip events, one possibility is to find the group of $N-1$ contact points with the lowest variance and determine the error influence normalized by this variance of the candidate slip point.

The slip events are used to window the traction force data. For each recognised slip event a window of M samples is taken from the beginning of the slip event. Three different data streams are considered: the traction force, the body linear velocities (from the odometry model) and the body angular velocities (from the AHRS). For each of the data streams a histogram is generated, and the three histograms are normalized and concatenated into a single feature vector. A support vector machine (SVM) classifier is used in order to determine if the feature vector belongs to a certain terrain class or not. Each terrain type has its own SVM, which generates a value from 0 to 1. The SVM is trained off-line from test data. Note that the classifier will only find conditions comparable to the training data. A change in environmental conditions which affect the slip properties of the terrain (e.g. wet grass) will likely result in poor classification results.

$$T_{\text{embodied}} = \begin{pmatrix} t_{\text{terrainA}} \\ t_{\text{terrainB}} \\ \dots \\ t_{\text{terrainN}} \end{pmatrix} \tag{2.40}$$

Analogous to the visual terrain classification (2.39), the result of the embodied classification process is a vector T_{embodied} of the same size, where each coefficient t corresponds to a classification value from the SVM classifier for a particular terrain in the range of 0 to 1. Like in the visual case the classification vector is binarised so that the terrain with the highest classification value is set to 1 and the other terrain types are 0.

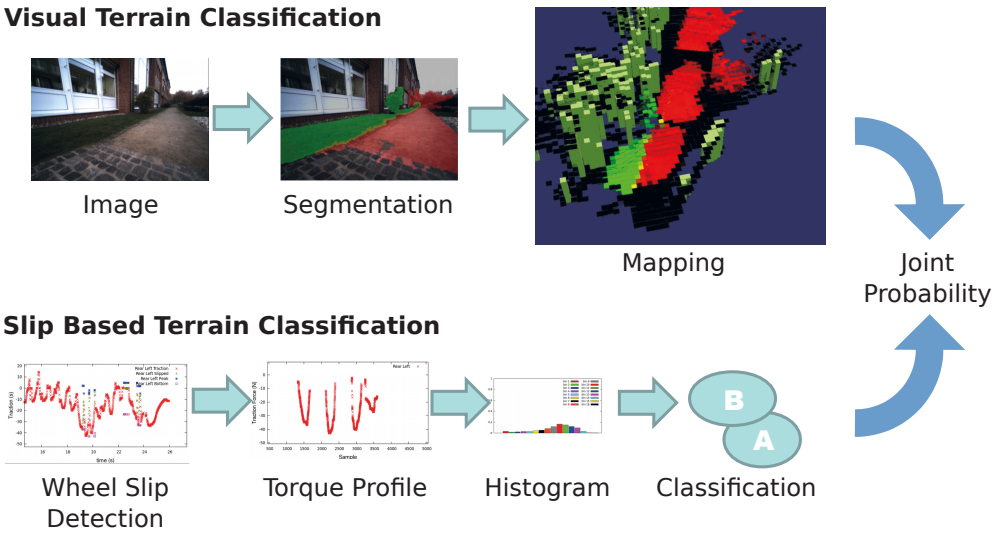


Figure 2.15: Processing chain for the joint terrain classification probability.

2.5.3 Terrain Classification Joint probability

In Section 2.5.1 and Section 2.5.2 we presented how to perform terrain classification based on both visual and embodied data. This data can be used to perform association of these two modalities. Figure 2.15 shows the processing chain for the terrain classification joint probability.

The visual information is first projected onto the existing map. This requires a known translation from the centre of the camera (pinhole model) to the body centre of the robot C^B_{CAM}. For each of the camera pixels, a map cell is identified which lies on the projective line through the centre and the pixel coordinate of the image pixel. If one or more map cells are intersecting with this line, the one closest to the camera centre is used for updating. A simple update method is used, which merges the classification array associated with the map cell T_{cell}, with the one from the visual classification T_{visual} (2.39), by performing an averaging between the two.

$$T^{k+1}_{\text{cell}} = \frac{T^{k}_{\text{cell}} + T_{\text{visual}}}{2} \tag{2.41}$$

Conversely, if a slip event occurred, the leg which was involved in the slip event is used for embodied classification of the terrain, such that the measurement z_k for the slip based model is the terrain vector T_{embodied} (2.40). The contact information

with the environment is given by the contact point c from the set $\mathcal{C}$, which is most likely for the leg where the slip event occurred (see contact point estimation in section 2.3.1). Based on the position of the body, and this contact point c (in body coordinates), a map cell is identified using the map function $m(C_{B_k}^W c, l)$ where the search interval l is three times the standard deviation of the uncertainty of the z position of the body and $C_{B_k}^W$ the transform from body to world given by the state x_k (see (2.13)). The likelihood of the slip based classification on the map cell, given the prior T_{cell} associated with that cell is calculated as

$$p(T_{\text{embodied}}|T_{\text{cell}}) = \begin{cases} p_{\text{correlated}} & |T_{\text{embodied}} \cdot T_{\text{cell}}| = 1 \\ p_{\text{uncorrelated}} & |T_{\text{embodied}} \cdot T_{\text{cell}}| < 1 \end{cases} \tag{2.42}$$

The values $p_{\text{correlated}}$ and $p_{\text{uncorrelated}}$ are fixed size values. For most of the experiments we have used 0.9 and 0.1 respectively. Given the definitions of the state x_k and the measurement z_k for this model, the likelihood can be put into the general form of (2.18) as

$$p(z_k|x_k, m) = p(T_{\text{embodied}}|T_{\text{cell}}) \tag{2.43}$$

Chapter 3

Particle Filter based Localisation and Mapping

In Chapter 2, the different models for the environment representation, as well as the prediction and the correction steps of the Bayes filter formulation in the context of embodied data associations have been discussed. The content of this chapter is to use these models in a Rao-Blackwellized particle filter (RBPF), which was introduced in Section 1.2.4. Section 3.1 describes a localisation setup, and introduces the state space representation used for the rest of the work. This localisation filter requires a map of the environment. That requirement is removed in Section 3.2, where a SLAM setup is described which generates 3D maps. The mapping is performed using optical sensors, while the data associations for the map correction are the same as in the localisation filter case.

3.1 Embodied Localisation

The full state space of the system has six degrees of freedom (DOF). Three for the translation in $\mathbb{R}^3$ and three for the rotational component in $SO(3)$. As stated in the introduction Chapter 1, the general Bayes filter requires an approximation of the probability distribution of the state space in order to be computable. The linear nature of the Kalman filter is in this case not a good approximation. Firstly because $SO(3)$ is non-Cartesian, and secondly because the visual-embodied correspondence

models are very non-linear and described in a piecewise fashion through the map function. Near to a wall for example, the contact based measurement model would exhibit a severe jump in the measurement likelihood. A much better approximation for the probability density function over the state space can be achieved by using samples. The problem only is that sampling over 6-degrees of freedom requires a very large number of samples in order to achieve sufficient sample density. Even when assuming that 10 samples are sufficient to cover one dimension – which they may not be depending on the spread of the probability density function (pdf) – this would require one million particles. The best approach to tackle this problem is to reduce the number of dimensions which need to be represented by samples.

The state of the system x_k was defined in (2.13) as $(x, y, z, \psi, \phi, \theta)$, with z being the axis parallel to the gravity vector, and ψ, ϕ, θ representing the roll, pitch and yaw axis. The roll and pitch axis can be estimated fairly well by the AHRS. Short term changes in these dimensions is measured by a gyro, while the long term stability and referencing is performed by an accelerometer. For this reason, the state estimation can directly use the values for the roll and pitch axis, with a fixed error distribution. How this error distribution looks like is part of the characteristics of the AHRS system. Current systems like the XSens MTi have an error of around 1° on these axis. In a terrestrial environment the AHRS can also give an estimate for the yaw axis. However, the long term stability of this axis is provided by magnetometers, which measure the earth magnetic field vector. There are multiple problems with this. Firstly, this will only work in a terrestrial environment, and many of the challenging scenarios are for lunar or planetary autonomous exploration (Haarmann et al. [2012]). The second problem with using magnetic sensors is that for ground based vehicles, there are many situations where the local magnetic field is altered due to ferro active structures in the vicinity. These disturbances will introduce a constant bias in the yaw estimation. While this is not preferrable in general, this bias can also be a useful alternative source of information for localisation (Le Grand and Thrun [2012]). Estimating the full yaw angle and estimating the bias of the AHRS is technically nearly equivalent. Therefore, sampling is used to represent the probability distribution of the yaw angle error in this thesis. The x and y axis also need to be sampled due to the non-continuity of the measurement functions on these axis. The z-axis is different

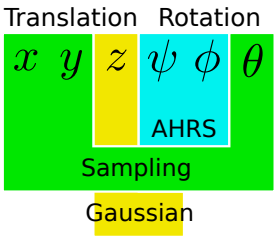


Figure 3.1: In general, six degrees of freedom are required to describe the pose of a robot in a 3D environment model. Using a sampling representation over all six degrees is computationally unfeasable with current processing capabilities. With the aid of an AHRS, two rotational degrees can already be estimated up to a fixed error. For ground based vehicles a Gaussian distribution can be used to approximate the z-axis, so that only three axis remain to be represented by samples.

again. Because of the assumption that the vehicle is in contact with the ground all the time, and the ground is also modelled as a Gaussian distribution, it is feasible to model the uncertainty distribution over the z-axis as a Gaussian. Mixing Gaussian and sampling representation has already been discussed in Section 1.2.4. Each of the particles which represent the sampling over x, y and θ thus carries a Gaussian parametrization over the distribution $\mathcal{N}(z, \sigma_z^2)$ of the z-axis. See Figure 3.1 for an overview on the partitioning of the state vector.

The goal of the filter is to approximate the posterior distribution of the state $p(x_k|z_{1:k}, u_{1:k})$, where z are the measurements and u are the control inputs to the system. This is done by chosing N particles which are drawn from that distribution

$$x_k^{[n]} = [x, y, \theta, z, \sigma_z^2], \tag{3.1}$$

where $n = 1..N$ is the index of a single particle. The set of all the particles for a timestep k is denoted as χ_k.

The initial state of the particle filter depends on the existing knowledge about the position x_0 of the system with relation to the map m. Lets say for now that we have a guess of the initial position $\hat{x_0}$ with a covariance $\hat{\Sigma_0}$. Then the initial particle set χ_0 is drawn from this distribution.

The sampled particles are forward projected using the odometry model update (2.15) from Section 2.2 for each new AHRS estimate. The practical update

frequency for the odometry model is 100 Hz. This moves the particles according to the mean odometry and spreads them so that they incorporate the uncertainty introduced by the odometry model. So, for a set of particles χ_k at time k, by transforming them through the odometry, we get a set $\bar{\chi}_{k+1}$, which can be calculated from (2.15) by taking the original sample and adding an odometry delta in the body frame of the particle. For this, the odometry delta has to be rotated by the θ of the respective particle around the z-axis.

$$\begin{aligned} \Delta_O &\sim \mathcal{N}(\mu_O, \Sigma_O) && (3.2)\\ \bar{x}_{k+1}^{[n]} &= x_k^{[n]} + \left[(R_z(\theta)\Delta_O)_x, (R_z(\theta)\Delta_O)_y, R_z^{-1}(\Delta_O), (\Delta_O)_z, (\Sigma_O)_z\right] && (3.3)\\ \bar{\chi}_{k+1} &= \{\bar{x}_{k+1}^{[n]}\} && (3.4) \end{aligned}$$

The subscripts $()_x$, $()_y$ and $()_z$ after the parenthesis denote the x, y and z component of the translation of the transform. $(\Sigma_O)_z$ is the covariance matrix of the odometry with all components other than z marginalized out, which makes it single variate. Note that Δ_O is drawn independently for each individual particle.

The odometry update step will always increase the particle spread because of the added error from Δ_O. Since the particles are an approximation of the pdf of the current state, this is equivalent to increasing the uncertainty of the state. To decrease the uncertainty, one ore more of the measurement models from Chapter 2 are applied to the particles (see Figure 3.2 for an example). This will not reduce the particle spread, but rather update the importance weights of the particles. The spread is reduced in the resampling step, which is discussed below. The weight of the n-th particle at time k is denoted by $w_k^{[n]}$. The weights are always kept normalized, so that the sum of all weights equals to unity. The measurement models all provide a likelihood of a measurement z_k given the current state x_k and the map m. So, according to (1.36) in Section 1.2.3 the particle weights are updated using

$$w_{k+1}^{[n]} \propto w_k^{[n]} p(z_{k+1}|x_{k+1}^{[n]}, m). \tag{3.5}$$

How z_k is defined depends on the specific model.

The update rule for the contact model from Section 2.3 is a special case for multiple reasons. Firstly it provides an estimate on the z-height $\bar{\xi}$ (2.27) and its

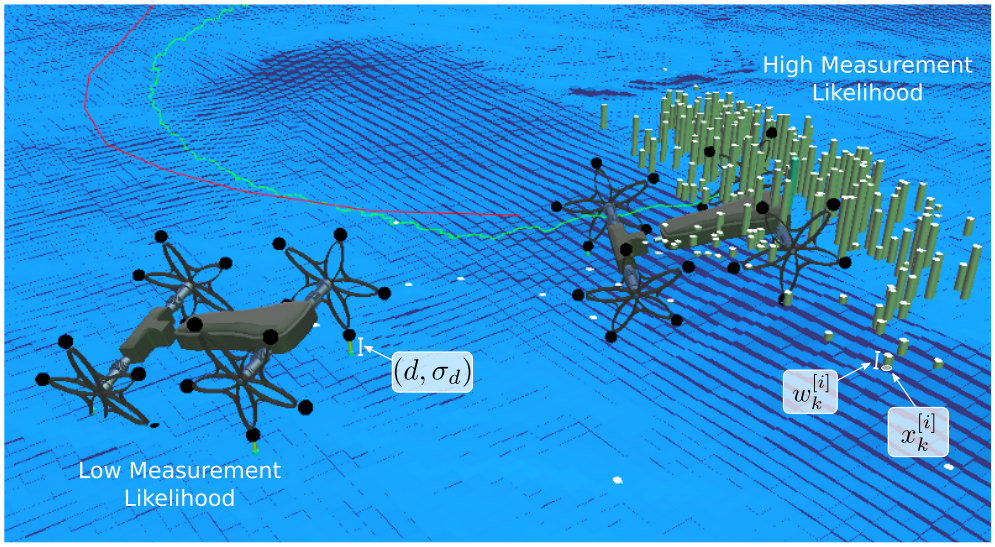


Figure 3.2: The localisation filter uses different measurement models to evaluate the likelihood of individual samples. The gray bars represent the particles and their relative weight. Two of the particles are highlighted and show the system and parameters of the contact based measurement model. For the particle on the left, the contacts do not fit the map very well, as can be seen by large values for (d, σ_d). The particle on the right has a better fit, as do the other particles in the vicinity.

uncertainty σ_ξ (2.28). So, in addition to (3.5) the z and σ_z values of the specific particle are updated with $(\bar{\xi}, \sigma_\xi)$ according to the Kalman-filter update rule (1.22). The second reason the contact model is special is the fact that the number of contact points within the individual particles is not necessarily constant. So, the measurement likelihoods are not comparable directly. This was compensated for by introducing the discounted average value $\nu\bar{p}$ in (2.26). The question is now how to calculate $\bar{p}$. As it is actually used as part of a product, it makes sense to take the geometric mean for each of the unnormalized measurement probabilities (2.25).

$$A = \{\hat{p}(z_k|x_k^{[n]})^{(1/|Z_k|)}|n = 1..N, |Z_k| > 0\} \tag{3.6}$$

$$\bar{p} = \sum_{a \in A} \frac{a}{|A|} \tag{3.7}$$

With a discounting factor $\nu = 1$, particles with no contact information would be

assigned an average weighting. Usually it is helpful to assign a factor $\nu < 1$ in order to favour measurements with more contact points over those with less.

So far the measurement models only update the particle weights, but not the distribution. Over time this will lead to a reduction in particle density, up to a point where the density is not sufficient anymore to adequately approximate the pdf of the state. The effective number of particles N_{eff} (1.38) is a measurement on how well the particles represent the underlying distribution (Gustafsson et al. [2002]). The particles are most effective if the relative difference between weights is low. If most of the weight is represented by only a low percentage of particles, the representation becomes ineffective. If N_{eff} falls below a threshold value, a resampling step is performed, which draws a new set of samples from the given distribution. This procedure will ensure that regions with high-weight particles (i.e. the regions where the actual robot is likely to be) will have a higher particle concentration after the resampling step. The are multiple schemes for sampling from the importance distribution. In this work both multinomial resampling as well as stratified resampling is used.

One detail of the algorithm, which has not been discussed so far is when to apply the measurement steps. In principle, the measurements can applied whenever they are available. However, most of the measurement models used here can be considered derived meausurements. For example the contact model uses AHRS and motor encoders to estimate how well the current configuration matches the terrain. To ensure at least some uncorrelatedness of consecutive measurements, the measurement step should not depend on when the AHRS and encoder readings come in, but when the distance reading to the map are likely to change. For this reason, the odometry readings are accumulated, and the measurement step applied whenever the odometry has moved more than m_{trans} or rotated more than m_{rot}. This also improves the runtime performance of the filter. The threshold values should be chosen so that on average new measurements are taken when the grid cell have changed. Without this step, due to the correlation of the measurements, the distribution for a non-moving robot would converge to the most likely particle and eliminate all other possibilities.

The following algorithm shows a pseudo-code implementation of the filter described so far. The filter used is a variant of the Sequential Importance

Resampling filter (SIR) (Gordon et al. [1993]; Gustafsson et al. [2002]).

Algorithm 1 Particle Filter

1: $D_O = I$
2: $x_k^{[n]} \sim \mathcal{N}(\mu_0, \Sigma_0) \forall n = 1..N$
3: **while** running **do**
4: $\quad \bar{\chi}_k = \chi_k = \emptyset$
5: $\quad D_O = \mu_O D_O$
6: $\quad$ **for all** $n = 1 \dots N$ **do**
7: $\quad\quad \Delta_O \sim \mathcal{N}(\mu_O, \Sigma_O)$
8: $\quad\quad x_k^{[n]} = x_k^{[n]} + R_z((x_k^{[n]})_\theta)\Delta_O$
9: $\quad\quad$ **if** $R_z^{-1}(D_O) > t_{\text{rot}}$ or $|(D_O)_T| > t_{\text{meas}}$ **then**
10: $\quad\quad\quad$ **for all** measurement models **do**
11: $\quad\quad\quad\quad w_k^{[n]} = w_k^{[n]} p(z_k | x_k^{[n]}, m)$
12: $\quad\quad\quad\quad$ **if** contact model **then**
13: $\quad\quad\quad\quad\quad x_k^{[n]}(z, \sigma_z) = \text{kalman\_update}(x_k^{[n]}(z, \sigma_z), (\xi, \sigma_\xi))$
14: $\quad\quad\quad\quad$ **end if**
15: $\quad\quad\quad$ **end for**
16: $\quad\quad\quad D_O = I$
17: $\quad\quad$ **end if**
18: $\quad\quad \bar{\chi}_k = \bar{\chi}_k + < x_k^{[n]}, w_k^{[n]} >$
19: $\quad$ **end for**
20: $\quad \hat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \left(w_k^{[i]}\right)^2}$
21: $\quad$ **if** $\hat{N}_{\text{eff}} < N_{thr}$ **then**
22: $\quad\quad$ draw i with probability $\propto w_k^{[i]}$
23: $\quad\quad$ add $x_k^{[i]}$ to χ_k
24: $\quad$ **else**
25: $\quad\quad \chi_k = \bar{\chi}_k$
26: $\quad$ **end if**
27: **end while**

The filter algorithm gives a rough overview of the steps involved. Line 1 and 2 are initialization of the relative movement since the last measurement update D_O, as well as the initial particle distribution. D_O is updated with the relative change from the odometry at each runtime step in line 5. The for loop in line 6 iterates over all the particles. For each particle, a sample is drawn from the odometry distribution. Line 8 gives a short notation on how this sample is first rotated into the frame of the particle and then added to the particle. Line 9 checks if the

conditions are met for triggering a measurement update. This is the case when the system has moved or rotated above a threshold value. Then, all the measurement models which we want to use in the filter are evaluated, and the measurement likelihood used to update the particle weights in line 11. For the special case of the contact model, the particles z and σ_z values are also updated using the kalman update rule. Line 16 resets the variable for triggering the update. The resulting weight and particle is added to the update set in line 18. Line 20-26 determine if the particles need to be resampled. If the effective number of particles drops below a threshold, the particles are drawn with a probability proportional to their weight from the update set in line 22.

The particles represent the distribution over the pose of the robot. In order to use this information in subsequent processing steps, it is often required to have a single pose instead of a distribution. The particle centroid is used for that purpose.

$$C^W_{\text{CENTROID}} = \sum_{n \in N} w_k^{[n]} C(x_k^{[n]}) \tag{3.8}$$

This is assuming the particle weights are normalized to sum to 1, and the transform $C(x_k^{[n]})$ is the body to world transform given by the state $x_k^{[n]}$ for the particle $[n]$. Note that to calculate the transform the actual representation of $C(x_k^{[n]})$ is relevant, because of the non-linearities of the rotation. The most straightforward solution is to represent $C(x_k^{[n]})$ as a 6-vector where the rotation is encoded as a scaled axis of rotation (Pennec and Thirion [1997]).

The calculation of the centroid required no consideration of the marginalization used on the z-axis. However, when calculating the covariance of the filter distribution this can not be ignored. It turns out that the covariance of the centroid is the covariance of the mean points plus the mean of the z-variance of the particles.

$$\begin{aligned} \Sigma_{\text{CENTROID}} \quad = \quad & \sum_{n \in N} w_k^{[n]} C(x_k^{[n]}) C(x_k^{[n]})^T + \sum_{n \in N} w_k^{[n]} (\sigma_z^{[n]} e_z)(\sigma_z^{[n]} e_z)^T \\ & - (\sum_{n \in N} w_k^{[n]} C(x_k^{[n]}))(\sum_{n \in N} w_k^{[n]} C(x_k^{[n]}))^T \end{aligned} \tag{3.9}$$

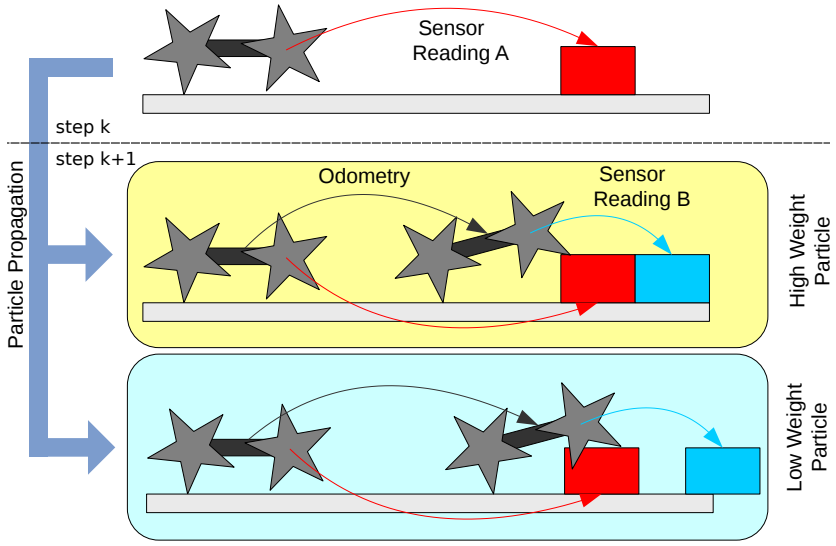


Figure 3.3: Each particle has its own environment map and z-position distribution. Sensor readings are added to the map according to the full particle position (top). The particles are propagated using the odometry model. A contact model determines how well the contact points of the robot match the map, and weigh the particles accordingly (center, bottom).

3.2 Local Map Generation

In the previous Section 3.1 a filter was described, which assumes the knowledge of the environment, and uses an odometry and the measurement models to estimate the pose of the robot. In this section, we extend the filter to cover the SLAM problem, where there is no a-priori knowledge of the environment. So in addition to estimating the probability distribution of the state x_k (2.13), we also estimate the map m.

Similar to the factorisation of the z-value in Section 3.1, Rao-Blackwellization (Doucet et al. [2000b]) is used to also factor out the map (Schwendner and Joyeux [2011]). The general principle of using one map per particle is illustrated in Fig. 3.3. Because of the nature of the problem, the map's posterior largely depends on the robot pose, as the pose error is the largest contributor to the map error. Since there is a surface on which the robot is in contact with, the component of the position parallel to the gravity vector (z) is closely connected to the map. For this reason, the full slam posterior is factored into the part which is represented

by a particle distribution

$$\bar{x}_k = [x\ y\ \theta] \tag{3.10}$$

and N maps including z-position

$$m^{[n]} = [z\ \sigma_z\ m] \tag{3.11}$$

which for the z-position is represented as a Gaussian and for the map as a regular grid structure in x and y (see Section 2.1). The full posterior can now be given by

$$p(\bar{x}_{1:k}|z_{1:k}, u_{1:k}) \prod_{n=1}^{N} p(m^{[n]}|\bar{x}_{1:k}^{[n]}, z_{1:k}), \tag{3.12}$$

where each of the N particles has a map and a z-position estimator.

With the factorisation, estimating the map and the z-position of the system is a subproblem which can be considered independently. So, effectively within the particle the state is considered true, which makes the map generation much more simple. The particles can be interpreted as different hypothesis. The weights will in the end determine which hypothesis will survive then next resampling and which ones will be eliminated. Within the context of the particle, the full position is provided by the position components x, y and θ from the particle as well as the orientation estimate from the AHRS. New sensor readings are added to the map based on this combined pose information. Special attention has to be given to the z-component of both position and map cells. This is because z is not sampled, but represented as a Gaussian distribution.

3.2.1 Optimal Estimator

One way of finding the map and z-position distribution $p(m^{[n]}|\bar{x}_{1:k}^{[n]}, z_{1:k})$ for a single particle is to use an information filter, where all the z-positions for each k steps and all map cells form the state of the filter. The information filter is the dual of the Kalman filter, only that it does not represent the uncertainty as a covariance matrix, but rather in its canonical form as a precision matrix. For a multivariate Gaussian this is the inverse of the covariance matrix. The results of both Kalman and information filter are the same. Only that the information

filter makes adding of measurements easy, and the precision matrix effectively stores a nice visual representation of the relation between the coefficients of the state vector. In the following paragraphs, a description of such an information filter is given. Note that this is mainly for understanding the principles and the approximations which are formulated later on. Therefore no full derivation of the filter is provided.

As mentioned, the information filter makes it simple to add measurements, but the prediction step is more complex. One possibility to get around this is to also view the odometry updates as measurements, instead of predictions. For this the state of the filter needs to hold not just the current z-position and uncertainty, but also all previous ones. The state of the filter can then be given as

$$s_k = [z_0 \ \dots \ z_k \, m_0 \ \dots \ m_n] \tag{3.13}$$

Note that m_0 to m_n are all the cells in the map. So for a grid of size $i \times j$ there are $ij + k$ coefficients in the state. The filter than has an information vector of the size of the state, and a precision matrix which is a square matrix where each side is the size of the state. Measurements are added to the filter and affect both the precision matrix and the information vector. Figure 3.4 shows an illustration of the precision matrix and how the measurements are added. Odometry measurements give the change in z-position and the uncertainty σ_z between two poses. They formulate the diagonal and adjacent fields in the upper-left submatrix. There are then two kinds of measurements on the maps which are both handled the same in the filter. Measurements of the environment are performed using exteroceptive sensors like Laser Range Finders (LRF) or Stereo Cameras. Each of these provide distance readings to elements of the environment. The measurements z_m are given in the reference frame of the sensor, and can be converted to the body frame through the $C^{\mathrm{B}}_{\mathrm{SEN}}$ transform. Since $C^{\mathrm{W}}_{\mathrm{B}}$ is known from the particle, and the only uncertainty is in the z-axis, the corresponding map cell can be identified through the mapping function m. For the information filter the measurement is provided as the relative difference between the pose and the map, so the z-component of the transformed measurement $(C^{\mathrm{B}}_{\mathrm{SEN}} z_m)_z$. This measurement is added to the diagonal elements of the lower-right submatrix in Figure 3.4 and the upper-right

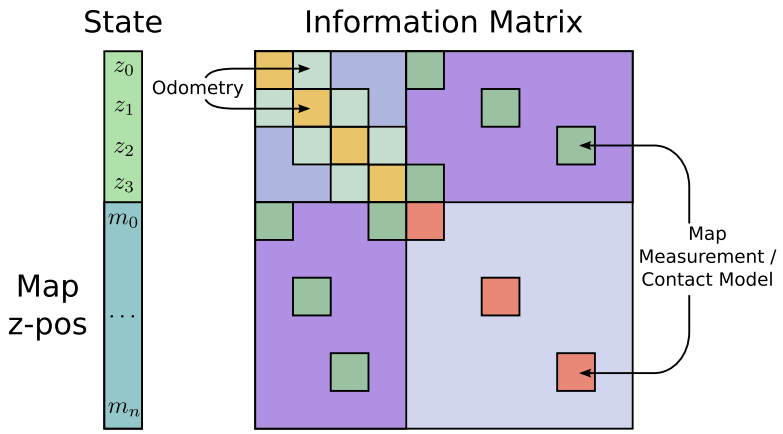


Figure 3.4: The information (precision) matrix represents the constraints between the ceofficients of the state vector, which are given by measurements. The state vector is a combination of the z-values of the robot pose as well as the map cells. The information matrix is used in the information filter, which is an optimal estimator of the given state. Note that an approximation of this optimal approach is used for this work.

(and lower-left) submatrix. The third type of measurement is when a the robot is in contact with a map cell that has already been sensed. For the information filter this is equivalent to the exteroceptive sensing of the map. The measurements which are added to the matrix and the information vector actually formulate constraints on the state. Lets say the information matrix at time k is I_k and the information vector i_k. The mean of the state $\bar{s}_k$ is then given by solution to the following linear system of equations:

$$I_k \bar{s}_k = i_k \tag{3.14}$$

To give an example of the process, a robot is moved in a grid world with 12×12 cells. The ground is flat, and the odometry as well as the map measurements have an uncertainty of $\sigma = 0.1$. The robot senses the four cells in front which are in its measurement cone. Figure 3.5 shows the mean estimated map height, which is equivalent to the map error since the true position is at 0. One can see that the relative odometry measurements lead to an increase in error over time. The overall error is reduced when the robot comes back to a position it has previously seen. This loop-closing reduces the overall error of the esimated map height. Similarly

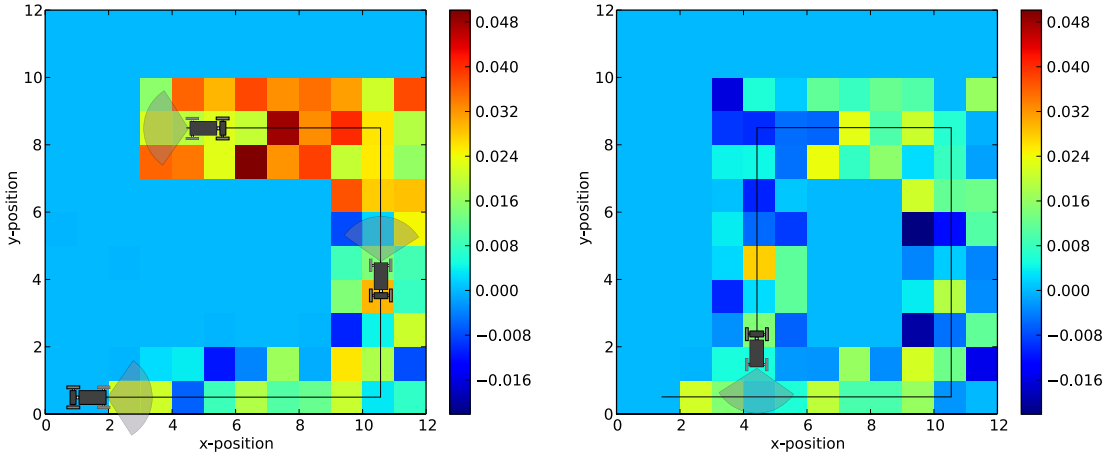


Figure 3.5: Simplified example of a robot in a grid world, to illustrate the optimal estimation of the z-height of pose and map. The map error is accumulated due to increasing pose errors over time (left). After the robot has returned to an already observed position, the z-error can be reduced over all parts of the map (right).

Figure 3.6 shows the z-error of mean estimate of the robot pose before and after loop closing. One can see that the error is linearly distributed backwards, once the loop has been closed. We can observe another important thing in this simple example. The size of the state vector is 35 steps $\times$ $(12\,\text{grids})^2 = 179$ elements. This results in an information matrix of 32041 cells. A dense representation of this matrix is thus not feasible for practical problems. Even when using a sparse representation, solving (3.14) will be too computationally intensive to perform for each particle.

3.2.2 Approximated map estimation

The matrix in Figure 3.4 already provides some clues on how to simplify the estimator. Firstly, the top-left submatrix, which relates the z-values of the poses only does so for consecutive elements. In this form, this is equivalent to summing up the z-axis mean and variances from the odometry. So that

$$z_k^+ = z_k + (\mu_O)_z \tag{3.15}$$

$$\sigma_z^{2+} = \sigma_z^2 + (\Sigma_O)_z \tag{3.16}$$

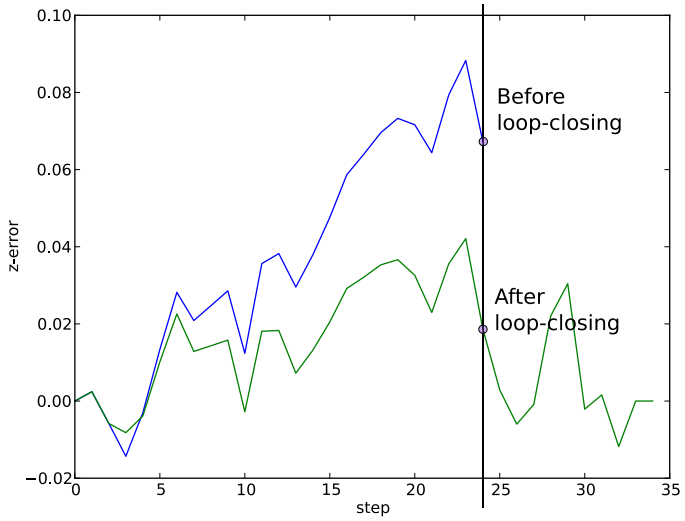


Figure 3.6: The graph shows the z-position error of the robot for the same scenario (but different run) as in Figure 3.5. After loop closing the error curve still has the same shape, but is adapted to fit the final position.

the next set of pose and uncertainty of the pose are based on the previous with the added z-components from the odometry. Equation (3.15) can be interpreted as the update equation for a single variate Kalman filter, which estimates the current z-position of the system.

Like in the optimal case for the information filter, two kinds of measurements are given also for the approximated solution. Exteroceptive measurements z_m are given in the sensor reference frame, and absolute position of the measurement in the W reference frame of the map is given by the following transformation chain:

$$p = C_{\mathrm{B}}^{\mathrm{W}} C_{\mathrm{SEN}}^{\mathrm{B}} z_m \tag{3.17}$$

the transformation $C_{\mathrm{B}}^{\mathrm{W}}$ is given by the particle, but only provides the mean for the z-component. The uncertainty is given by adding the uncertainty of the measurement to the uncertainty of the pose from which it was measured

$$\sigma_p^2 = \sigma_m^2 + \sigma_z^2. \tag{3.18}$$

The measurement pair (p, σ_p^2) is now used to identify the surface patch using (2.5) with $l = 3\sigma_p$, and updating it according to (2.4). Since (p, σ_p^2) gives the measurement in world coordinates and also include the accumulated error of the pose, the map of the particle approximates the one from the optimal case excluding error feedback from the map measurements.

Performing a full error feedback of the map measurements is quite costly to perform for each particle. For this reason, the method to incorporate this feedback here is through the use of the contact model from Section 2.3. Specifically the maximum likelihood measurement for the z-height $\bar{\xi}$ (2.27) and its variance σ_ξ^2 (2.28) is used as a measurement update to the Kalman filter, which estimates the current z-position of the system. The Kalman filter formulation requires that the measurement noise is independent. The problem is, that for directly using $(\bar{\xi}, \sigma_\xi^2)$ this is not the case. Remember that in (3.18) the position error is added to the measurement error, and that the same error is used in (3.15). Since $\bar{\xi}$ is based on the map heights around the position of the robot, the error up to the step at which the map was updated is fully correlated (see Figure 3.7 for further information). So instead of applying the Kalman filter update equation to the full error variance of the current pose, the correlated part is removed. For the mean this makes no difference, but the variance for the current estimate and measurement is reduced by the common variance σ_c^2 to calculate the gain and update, and afterwards added again, so that the Kalman update equation (1.22) is adapted to:

$$\bar{\sigma}_z^2 = \sigma_z^{2+} - \sigma_c^2 \tag{3.19}$$

$$K_k = \frac{\bar{\sigma}_z^2}{\bar{\sigma}_z^2 + \sigma_\xi^2} \tag{3.20}$$

$$z_k = z_k^+ + K_k(\bar{\xi} - z_k^+) \tag{3.21}$$

$$\sigma_z^2 = (1 - K_k)\bar{\sigma}_z^2 + \sigma_c^2 \tag{3.22}$$

One question that remains is how to get the common variance σ_c^2. With the assumption that the pose error is always increasing, we can provide an upper bound of the common error by using the variance of the z_u at the time u when the map cell was last updated. Therefore the cells of the map need to store the time-step at which they where updated. For the contact model, which uses

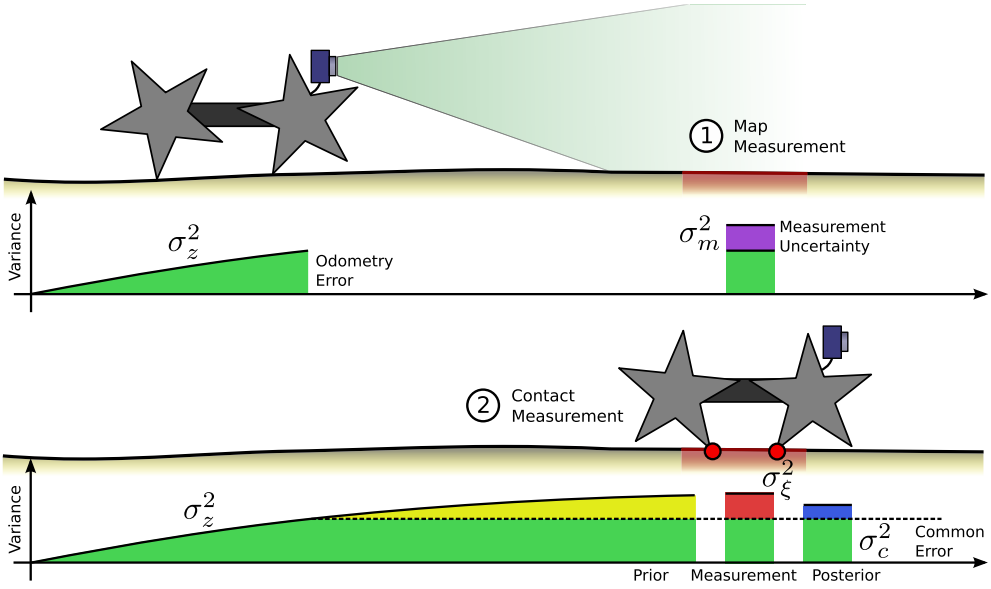


Figure 3.7: Map measurements (1) combine the z-error from the pose with the sensors measurement error. When the the contact measurement is used to update the estimate of the robot's z-position (2), it has to be taken into account that this measurement can only correct the error that was accumulated since the map was updated. This common error forms a lower boundary for the posterior estimate of the z-position.

multiple cells, the mean of those variances is taken.

To summarize the approximated solution: Instead of using the full information matrix between all states and all map cells, a single variate Kalman filter is used to estimate the current z-position of the robot, and the map model is used to estimate the map cells. In addition, each map cell stores an index of the step when it was last updated, and a list is kept which contains the variances of each timestep an update was performed. Comparing this approximation to the optimal estimation example from before, we now only have to store 179 values instead of the 32041 values. More importantly, no system of equations of this size has to be solved.

In order to verify the validity of the approximations a simulation was set up, which uses the Asguard model (see Section 6.1.1) on a flat surface. The robot travels for 10 m in a straight line and generates map measurements 1 m in front of it. The z-part of the current position determines the z-position of the map cell.

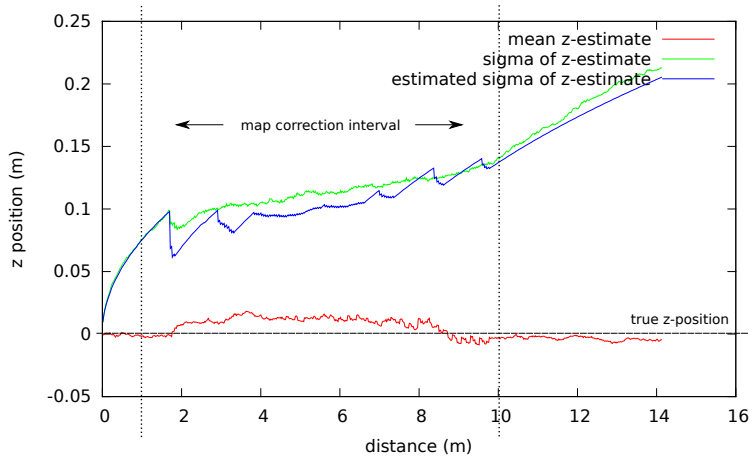


Figure 3.8: Simulation run averaged over 500 iterations, which shows statistics on the estimation of the z-position under the influence of the map correction. The simulated robot starts at distance 0 m and always has a true z-value of 0 m. Map surface patches are perceived at height 0 m between distance 1 m and 10 m. In this region the z-estimate is corrected by the map measurements and the variance grows more slowly. The estimation of the variance on the z-position matches the calculated variance of the z-estimate over all iterations.

Any error in the z-position of the robot is translated to the map cells. The position was perturbed by odometry noise, and map measurements by measurement noise. The simulation was run 500 times, and the estimated variances of the z-estimate compared to the sample variance. Figure 3.8 gives the averaged results of the simulation, which show that the approximations made are viable. The pose and map variance are approximated within tolerable boundaries. In this result, the position mean has a slight bias, which is due to the geometry of the Asguard system.

Chapter 4

Global Pose Constraint based SLAM

In Chapter 3, a particle filter based method was introduced, which is able to use visual-embodied data association to generate 3D environment models for ground based systems. One of the main problems with the sampling based approach of the particle filter is to maintain suitable particle density to represent the probability distribution over the robot pose and the map. The map error is closely related to the pose error, as all map measurements are taken from the position of the robot. Map measurement errors are therefore a sum of the pose error and the actual measurement error. The measurement error will most of the time be independent of the pose, and can be considered constant on average in this context. The pose error, as mentioned several times before, grows with travelled distance due to the uncertainty in the odometry. So the farther the robot travels, the more uncertain its pose becomes. A higher degree of uncertainty also means a more spread out probability density function and with constant number of particles a lower particle density. So the method from Chapter 3 becomes less suitable with increasing distance travelled.

Once previously seen places are revisited, it is possible to reduce the position error with respect to the original position. This is called loop closing, and has seen extensive research already (e.g. Montemerlo et al. [2002]; Sprickerhof et al. [2011]; Stachniss et al. [2004]). Particle filters can implicitly close loops by reinforcing

particles with higher global consistency (Montemerlo et al. [2002]). In practice this is a problem for larger runs, since the number of particles is limited by CPU and memory. When the uncertainty grows, the particle density is reduced, so that the chance that one of those particles represents the correct loop shrinks with growing distance. This is referred to as particle depletion (Doucet et al. [2000b]).

As already outlined in the overview, the approach taken in this work is to use a hierarchical approach to this problem (Schwendner [2012]). The sampling based method is used to generate locally consistent map segments. When the robot has travelled a certain distance, the state of the particle filter, including all of the poses and submaps, is stored as a map segment. Each segment has its own frame of reference. Once a segment has been stored, it is not revisited again. Two consecutive segments are connected in the sense that the start of the new segment is at the end of the last segment. The start of a segment is always at the origin of its local frame. The end of the segment is determined by the final positions of the particles. Since the particles represent a probability distribution, this constraint is also a probabilistic one.

The result is a chain of segments with probabilistic constraints between them (see Figure 4.1). So far there is no actual gain over the particle filter representation. The gain arises when constraints are generated which connect different elements of the chain and loop-closing can be performed. In this case pose constraint graph optimization methods can be used to arrange the local frames according to a least square criterion for the sum of all constraints. This effectively finds the optimal location of the segments with respect to a global reference in a least-square sense.

In Section 4.1 a method is described where constraints are generated using a stereo camera based approach which relies on sparse features to associate two poses. Any other method which can spatially relate two map segments could be used as well. One example which is used commonly in other works is the iterative closest point (ICP) algorithm. The visual approach was chosen here, since the false-positive rate for the matches can be controlled better.

Once the the map segments are arranged using the global optimization described in Section 4.3, the errors can be back-propagated (Section 4.4) into the local maps. This results in globally and locally consistent maps, which are suitable for the use in a navigation stack as described in Section 5.

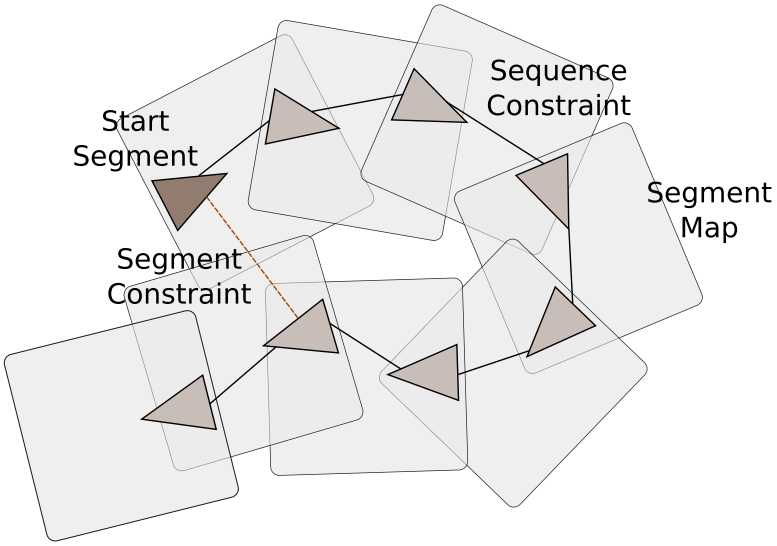


Figure 4.1: The hierarchical SLAM approach presented in this thesis uses local map segments generated by the RBPF approach from Chapter 3 and puts them into relation with probabilistic transformation constraints.

4.1 Visual Constraints

There are multiple possibilities to provide constraints between the local maps. This section describes a method which is based on matching image features from a stereo camera. Firstly, the features are matched between the calibrated left and the right cameras, in order to get feature points with a 3D position relative to the cameras. The SURF (Bay et al. [2008]) feature detector and descriptor are used for this purpose as they provide a good balance between matching accuracy and speed. The result is a set of 3D-points for which a relation to the coordinate system of the local map is known. Each of these points has an associated feature descriptor. The feature descriptor is a point in a 128 dimensional feature space. The characteristic of this feature space is that feature points that are derived from the same visual pattern have a low eucledian distance, regardless of their orientation and scale in the image space. So the distance between two feature descriptors can be used to estimate the likelihood that two different measurements represent the same 3D point.

This combination of 3D-points with associated features is stored with the local map segment and can be used to generate constraints between map segments. To

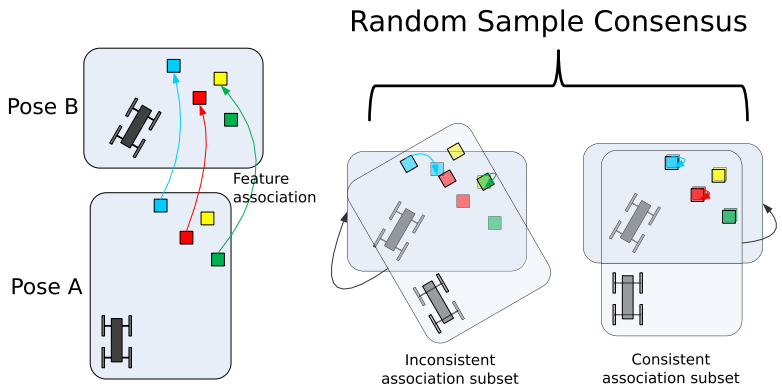


Figure 4.2: The feature matcher will produce a number of candidate pairs. The problem is that this set of pairs will also include wrong matches. A Random Sample Consensus (RANSAC) algorithm is used to find a subset of pairs which are geometrically consistent.

match two local map segments, the feature points from segment A are matched to those in B using a nearest neighbour search in feature space. Even when using thresholding on the distance and other heuristic methods to reduce the amount of found feature pairs, a large number of false matches remain in practical applications. The approach employed in this work is to use a RANSAC (Fischler and Bolles [1981]) algorithm to try to find a subset of the feature pairs, which fit an isometry constraint between the two feature clouds. This means that there must be a transform in this subset that transforms each point in A onto its corresponding point in B within a given error bound ϵ. The illustration in Figure 4.2 shows the principle of finding the geometric subset of features that fit.

If a subset of matching point pairs is found which is large enough to reduce chance matches, a constraint between the two local map segments is generated. In practical applications it was found that 7 point pairs is a suitable number of point pairs. The points in the pairs are each from a different coordinate frame. To associate the coordinate frames with each other, we have to find the transform which minimizes the sum of the square distance between the pairs. There are multiple methods to achieve this. Here we use the one from Besl and McKay [1992]. This transform forms the mean of a probabilistic constraint between two segments. The covariance is scaled based on the number of actual point pairs found.

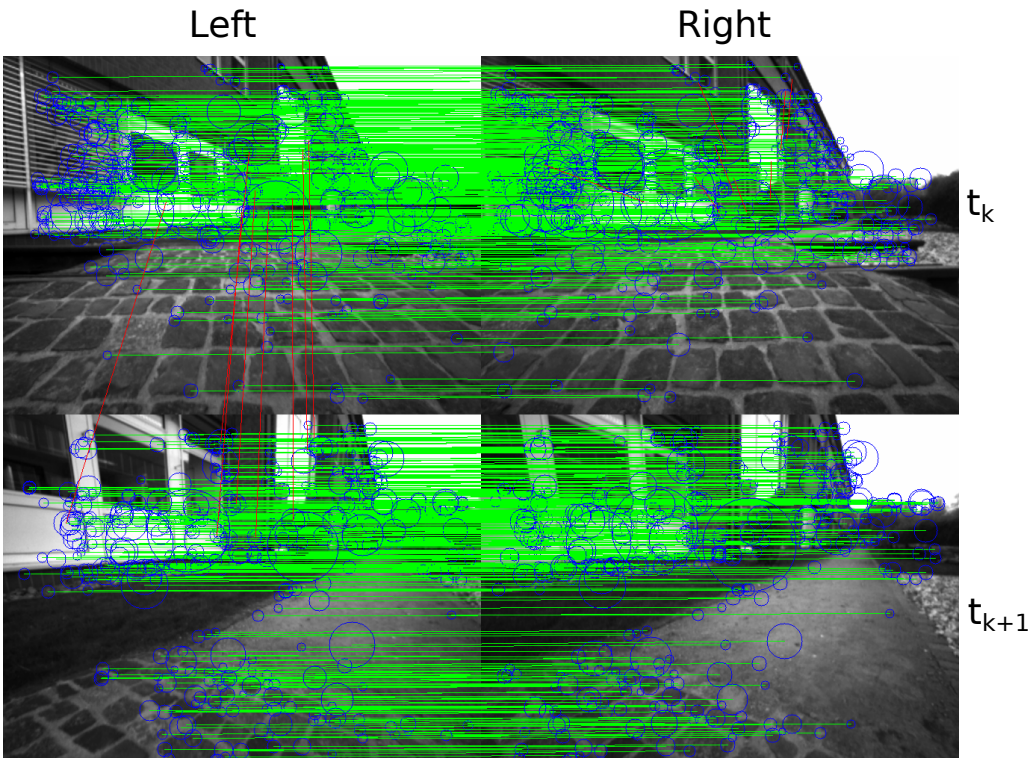


Figure 4.3: Two stereo feature pairs from a moving Asguard system with around 1.5 m translation of viewpoint. The blue circles are SURF features detected in the images. Green lines mark features that are matched using the epipolar constraint of the stereo camera setup. The inter-frame matches in red are a subset of all feature matches that where found using a RANSAC algorithm.

In Figure 4.3 an example image is shown for two stereo image pairs, for which a geometrically consistent subset of feature pairs was found. The distance in viewpoints between the two timesteps is around 1.5 m. Due to the low vantage point of the system over ground, a large portion of the actual image has changed and features in the near field can not be matched. For the actual match features in the mid-field are used instead, which results in a larger uncertainty of the match. Also, the features are not very resilient to changes in the perspective projection. For that reason only views which have been taken from a similar viewpoint can be matched in this way.

4.2 Segment association candidates

In Section 4.1 it is shown how to match two segments by associated feature clouds. The process of checking for constraints between two segments is usually fairly costly. The number of possible segment pairs is the square of the number of segments, since with each additional segment the number of possible matches to other segments grows linearly. For this reason, we use a heuristic based on the extents (bounding box) of the map content for each segment S_i. When the bounding box of two segments do not intersect, no true positive match between the segments is possible. The problem is only that the bounding boxes are expressed in the local frame, and the relation between the frame of the segment and the other segment is what we are actually trying to figure out. Due to the generation process of the segments there is always an estimate of the error between the local frame and the global frame. So to provide a proper heuristic, bounding box needs to be extended by the uncertainty with respect to the other frame. This does not have to be exact as it only provides an upper boundary for a decision to perform a proper matching step later.

The method employed in this work is to take the corners of the bounding box and apply the uncertainty transformation, effectively turning the point into an uncertainty ellipse. Then the sigma points which lie on the principal axis of the ellipse are used in a 3-sigma environment. The bounding box around all these sigma points now incorporates the uncertainty in the reference frame of the segment and can be used to decide if two segments should be considered for matching or not. Figure 4.4 shows an illustration of the concept. If the extends for S_i calculated with this method overlap with those from S_{i-1}, these two segments can be evaluated for correspondence.

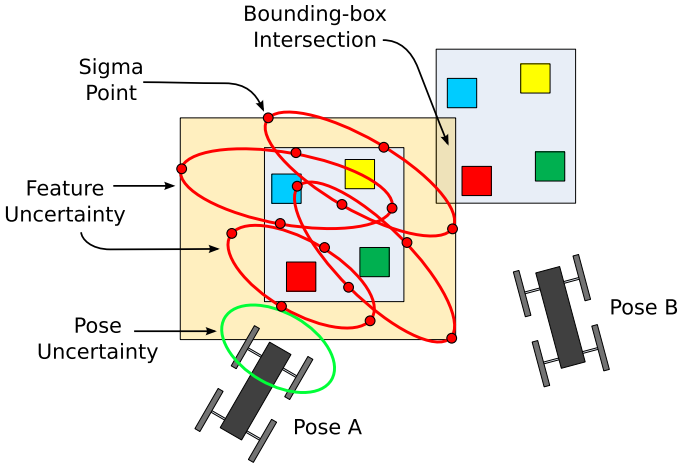


Figure 4.4: Matching two poses against each other is a costly operation, and should not be performed unnecessarily. The method used in this work to find out if two candidate poses and their associated may result in a match is to look for an intersection of the bounding box. The bounding boxes encompasses the features including their uncertainty ellipses according to the pose uncertainty.

4.3 Segment constraints graph

In the presented approach small-scale maps – called segments – are generated using the method described in Section 3.2 and combined into larger maps. Each segment has its own frame of reference, where the starting position for all particles is at the origin. Within the context of the reference frame there is no uncertainty on the starting position of the system. This is of course different to the uncertainty with respect to the world frame. The particle filter mapping algorithm is run until one of the particles passes over a certain threshold boundary in either x or y direction relative to the origin of the segment. Once the threshold is passed, the state of the filter is frozen in a segment representation

$$S = (T, \Sigma_T, p_{1:N}). \tag{4.1}$$

The segment holds the mean T and covariance Σ_T of the transformation from the final pose within the segment to the origin as well as the state $p_{1:N}$ of each of the N particles. T and Σ_T are extracted from the final particle poses and their relative weighting using (3.8) and (3.9). Using a Gaussian here is required in order

to be compatible with the graph optimizer, but is an approximation of the pose distribution represented by the particles. Some cases, like with multiple strong local maxima, may not be represented well using this approach. An alternative for future work would be to interpret the particles using a Gaussian Mixture Model. The state of each particle is recorded as

$$p_n = (T_n, w_n, m) \tag{4.2}$$

Which holds the transform T_n of the final pose of the particle to the origin of the segment, the weight w_n and the map m of that particle.

The individual segments are organised in a graph structure G. Segments are represented as vertices, and constraints between segments are the edges of the graph. Let $x = (x_1, \ldots, x_n)^T$ represent the pose x_k of each vertex with respect to a common frame W. The sum of all errors given by the constraints can be given as

$$F(x) = \sum_{<i,j>\in C} e_{ij}^T \Omega_{ij} e_{ij}. \tag{4.3}$$

A method like presented in Grisetti et al. [2010] can be used to find $x^* = \arg\min_x F(x)$ such that the error is minimized. The information matrix Ω_{ij} is the inverse of the constraint covariance matrix stored in the graph, while e_{ij} is an error function expressing the difference between the current pose transform between i and j and the measurement. See Section 1.2.5 for more detailled information on the graph based SLAM approach.

The first segment added to the graph G is added with an anchoring constraint to the origin of the common reference frame. Any consecutive segment S_{k+1} is added with the constraint that the origin of segment $k+1$ relative to the origin of segment k is (T, Σ_T). Apart from these sequence constraints, additional constraints can be added between S_{k+1} and any Segment S_i with $i \leq k$. The additional constraint added in this manner in this work is the visual loop closing constraint presented in Section 4.1.

Note that constraints can be added at any time of the process between any of the nodes. Each additional constraint will affect the optimal solution for the relative position of the map segments. The handling of constraint outliers is not

covered in this thesis, and the reader is referred to other works like Olson [2008] instead.

4.4 Error back-propagation for final map

The pose graph G now provides all the necessary information in order to generate a current best estimate of a global map. After running the pose graph optimizer on the graph, the optimal estimate on the transformation x_i from the frame of segment i to the common reference frame is available. The optimal relative transformation between two consecutive segments is given by $x_i^{-1}x_{i+1}$. Since all segments have been generated in a chain, and given that the uncertainties have been appropriately modelled, this relative transformation provides a clue on how to select the optimal map fit of that segment. It is effectively a best guess on the final pose of the system within a particular segment. With this information we would now like to extract a map from the segment which incorporates this information. Because of the marginalization of the state this is handled in different ways depending on the axis. For the x, y and θ axis we have a sampling representation, which represents a complete trajectory. For each particle in a segment S we can calculate the error to the optimal transformation as

$$e_n = |T_n - x_i^{-1}x_{i+1}|. \tag{4.4}$$

where T_n is the final pose of the particle from (4.2). The map m_n of the particle with the smallest error e_n is selected as an approximation of the optimal map for that particular segment. Figure 4.5 shows a visual representation of this optimal map segmentation for an optimized graph.

Because we have used a different representation for the z-component of the map, the selection of a particle does not automatically propagate the error of the z-axis back onto the map. Remember Figure 3.6, where the the z-error was back propagated through the information filter after loop closing. The same principle is now applied for the selected map m_n. Let's say that

$$e_z = (T_n - x_i^{-1}x_{i+1})_z \tag{4.5}$$

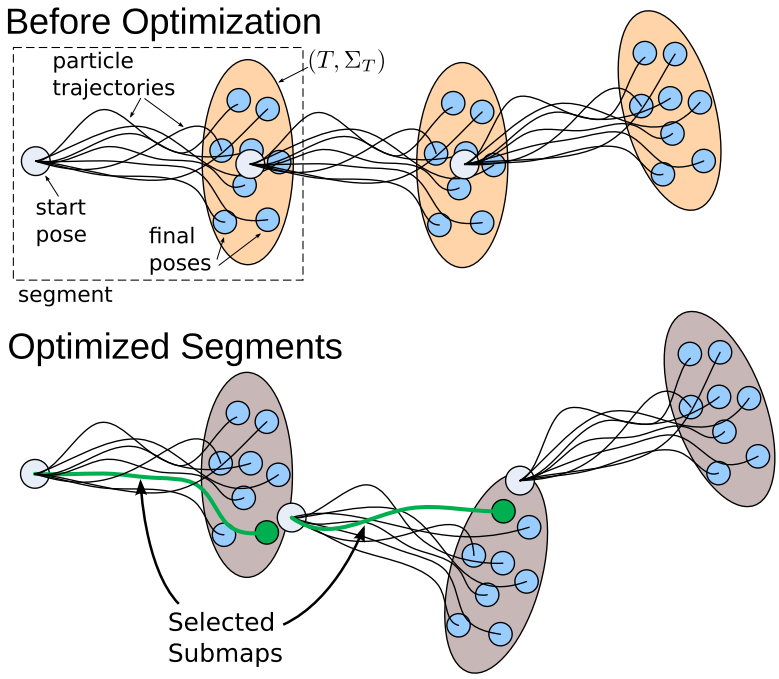


Figure 4.5: Individual map segments consists of the particles and their associated trajectories and maps. Constraints between segments are modelled as the normal distribution over the weighted particle poses. After including other constraints e.g. from loop closing, the relative alignment of the graph is changed. The final map is evaluated from the map pieces, which have the smoothest transition to the next segment (green).

is the z-component of the pose error. We would now like to linearly propagate this error into the map as visualized in Figure 4.6. Each map cell might have been updated from different poses. Because of the approximations taken in the previous chapter to make the map generation computationally feasible, we have only stored the index k of the last pose when the cell was updated, and t is the number of total pose indices used in the segment.

The mean m_μ for each map cell is therefore updated as

$$\hat{m_\mu} = m_\mu - e_z k/t. \tag{4.6}$$

After this update step, we have a collection of concrete submaps, each in their own frame of reference. These submaps can be combined in a single map, by applying the local reference frame transformations to the center of the individual

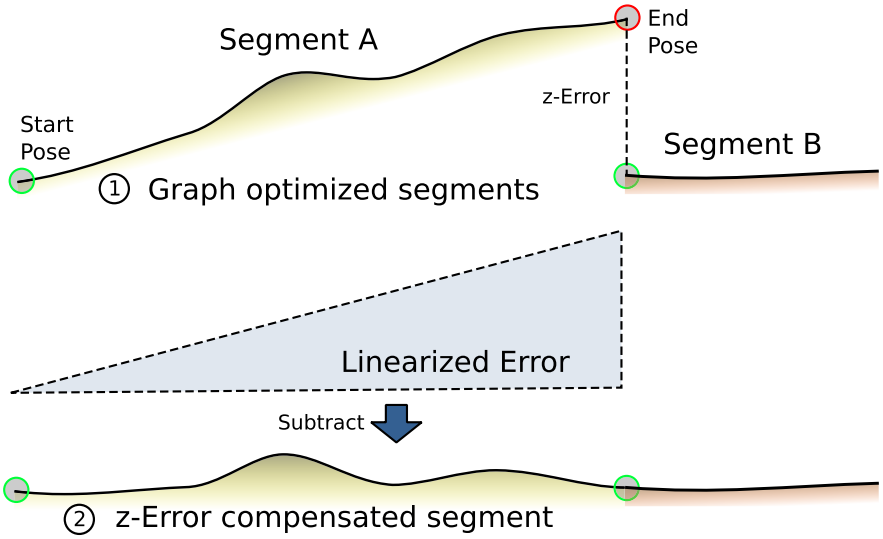


Figure 4.6: The x, y and θ axis are sampled and error back-propagation is performed by selecting the appropriate sample. The remaining error on the z-axis between two segments (1) is linearly distributed back over the map. Each map cell is compensated with a fraction of the final error proportional to the time step when the cell was last updated (2).

map cells. This results in some aliasing issues, but proved suitable in practical applications. The process of generating the global maps is summarized in Figure 4.7.

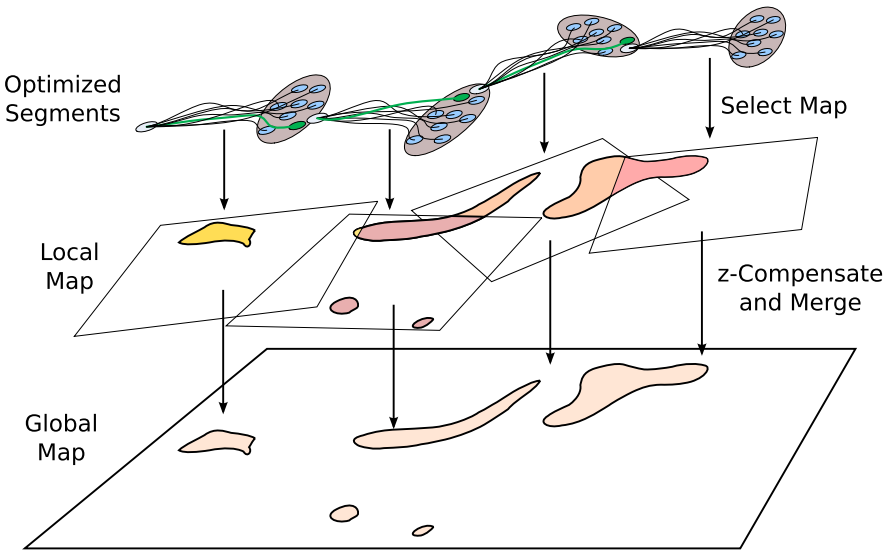


Figure 4.7: This illustration shows the process of generating the global maps based on the local segments and the constraints. After optimization of the constraint graph, the optimal maps for each segment are selected, the z-error is compensated, and the individual maps merged into a global map.

Chapter 5

Integration into navigation solution

In Chapter 2 to Chapter 4 a method was described, which uses embodied data in order to perform localisation and map building for ground based robots. For practical systems the generation of maps is not an isolated process, but rather embedded in a larger navigation system. The purpose of this chapter is to describe how the localisation and mapping method of this work can be embedded in a real navigation solution for the example of the Asguard (Section 6.1.1) system. The navigation stack of the Asguard system has been developed as part of the "Intelligent Mobility" project which was funded by the Federal Ministry for Economics and Technology (BMWI) through the German Space Agency (DLR) grant number 50 RA 0907.

While the work on the embodied SLAM method described so far has been performed solely by the author, the design and implementation of the navigation stack described in this chapter is a collaborative effort of the Intelligent Mobility team. The purpose of this chapter is to show how the SLAM solutions which have been proposed in this work can be used in a real working system, and provide some insights on the technical aspects which are involved.

The first Section 5.1 describes the general scenario to provide a motivation for the description of the technical solution in Section 5.2. Further, a short introduction of the environment representation library is given in Section 5.3.

5.1 Exploration Scenario

The principle scenario of the Intelligent Mobility project is the navigation of a ground based system in an unknown, unstructured and difficult outdoor environment. No GPS is available to aid the navigation. An example of such an environment would be given by a lunar exploration mission like described in Schwendner et al. [2009] or Haarmann et al. [2012]. A fully autonomous system needs to travel for a distance of around 100 m and return to the origin. The system has no knowledge on how the environment looks like and is required to avoid any kind of obstacles. In this scenario, the environment is assumed to be static. The general structure will remain constant for the duration of the mission. The system is equipped with a locomotion subsystem which is in principle able to traverse the given terrain, but not the obstacles. Operation is assumed to be fully autonomous, and all processing which is required needs to be performed on board. The energy for operating the system is also not supplied externally. The system has a sensorial set-up which is able to gather 3D information on the surrounding of the system. This can either be a Laser Range Finder (LRF), a 3D camera or a stereo camera set-up. Further, the system requires to have an Inertial Measurement Unit (IMU) with an AHRS as well as the ability to sense where its principal points of contact with the environment are situated.

5.2 iMoby navigation stack

The implementation of a complete autonomous navigation stack require different components to work together and exchange information. Further, different operational modes may require a reconfiguration of the data flow between the components during run-time of the system. The technical implementation of the iMoby navigation stack is based on the Robot Construction Kit (Rock), which uses Orocos as its underlying component middle-ware. The advantage of Rock over alternative component and robot integration frameworks is the ability to explicitly model the relationships between the components and provide a generic runtime management of the components and their interactions. The description for the navigation stack given here is however completely independent of the Rock

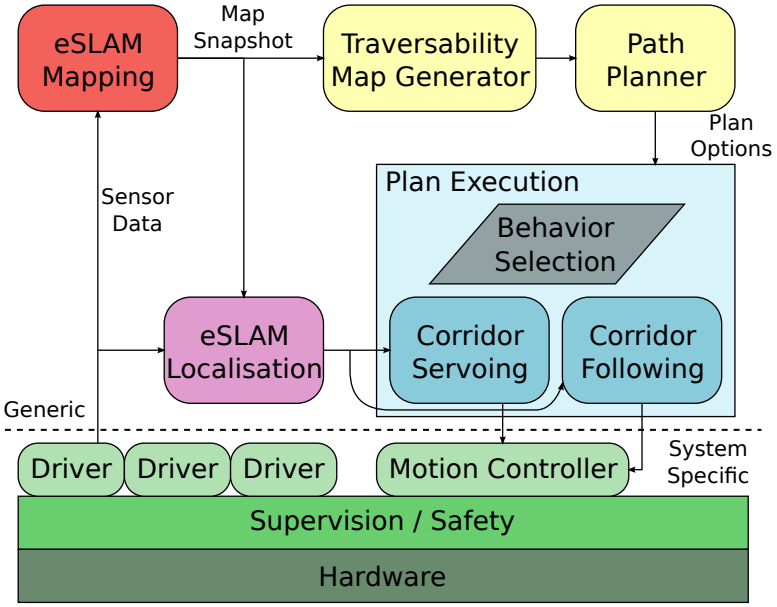


Figure 5.1: High-level components of the iMoby navigation stack. The lower part is specific to the actual system, while the upper modules are system independent.

framework and may well be implemented in other frameworks with similar feature sets.

Figure 5.1 shows an overview of the components which are involved in the navigation stack. At the bottom is the robot hardware and low-level supervision and safety system. This layer ensures safe operation on a hardware level. This includes things like protection against over- and under-voltage of the power supply, thermal protection as well as remote safety switches. On top of that is the robot specific hardware layer. This includes a motion controller, which can transform motion commands like *forward* or *turn* into robot specific commands for the individual actuators. Also the sensor and hardware drivers are in this layer. Note that the two systems which have been used for evaluating the eSLAM method have completely different implementations of this layer (see Section 6.1.1 and Section 6.1.2).

The aim of the navigation stack is to operate in an unknown environment, so whatever the system does, a continuous mapping process is required in order to build a model of the environment. The mapping block shown in Figure 5.1 constitutes the filter described in the previous chapters. It requires the sensor in-

formation from the AHRS, the environment contact points, 3D sensor information as well as the feature-clouds for performing association of submaps. It continuously holds a representation of the environment including the uncertainty over the local maps and the global constraints. This information comes from different modalities, and is stored in a unified way described in Section 5.3. The mapping task can receive a trigger input to optimise the currently known constraints and provide a current best map given the sensorial information so far as well as the perceived position of the robot within this map. This process usually requires some processing time in the order of a few seconds and thus is not performed continuously, but only when a new map is required. One important aspect to note is that the map which is generated can drastically change between different time steps. This is due to the fact that internally multiple hypothesis are tracked, and depending on the updated sensorial data these hypotheses may alternate strongly. This property is going to be quite important for other aspects of the system described later on.

The second goal of the stack is to navigate from A to B within the given environment. For this reason we assume that a point B is provided with relation to the start point A. At any given time, including the start, where the robot is uncertain what to do, the path planning loop is triggered. The first step of this loop is to trigger the generation of a current best map from the mapping module. This map is used as the basis for operation until the planning loop is triggered again. There are two different pathways for this map. The first pathway is for the generation of a path plan. Path planning seeks to find a way through the environment model which is in principle traversable for the system. In order to assess if the robot can negotiate certain parts of the terrain, a traversability map is generated. The basis for this map is the 3D environment model. Each cell on the plane of this 3D model is classified based on several factors like slope and adjacent cell height. Different classes relate to different costs for traversing this cell. For example steeper cells are more costly than flat cells. Map cells which are not traversable are classified as obstacles. Once the traversability map is classified the obstacle class is grown by half the robot's width, in order to close small corridors which would not actually be traversable by the system. The 2D traversability map is then used in the path planner module. The path planner of

the iMoby stack is a near-optimal planner. The difference to an optimal planner is that instead of generating an optimal path to the goal based on a given cost function, it generates areas which are near optimal (Joyeux and Kirchner [2009]; Joyeux et al. [2011]). From this area multiple candidate corridors are extracted which serve as alternatives for the plan execution part of the system. For example the plan execution might decide to choose a wide corridor over a narrow corridor or use other criteria to make a decision which is independent of the actual cost function which was provided to the plan generation. Once the plan is generated, it is passed to the plan execution subsystem. This system selects the plan options and appropriate behaviours which need to be performed in order to fulfil the plan.

In the iMoby stack two different execution behaviours have been implemented. The corridor following (Figure 5.2) behaviour uses no sensor feedback for navigation and relies on the localisation subsystem to provide the required information to navigate the path. Once the uncertainty of the localisation goes beyond a safe distance to the border of the corridor, the corridor following behaviour stops and reports to the supervision system that it can not safely operate in the given plan. The supervision system in this case decides what to do. The second behaviour which can be selected by the plan execution subsystem is the corridor-servoing (Figure 5.3). This behaviour takes the sensor information in order to generate a map of the direct vicinity of the system and performs motion planning in the general direction of the provided corridor. This behaviour is robust to obstacles in the path and can compensate high uncertainties in the localisation. The additional abilities come at the cost of higher computational power requirements. Similar to the corridor following, the corridor servoing behaviour stops and reports a failure when no trajectory can be found in the general direction of the corridor. Both locomotion behaviours generate trajectories, which are passed to the locomotion layer of the navigation stack. The trajectories are transformed into locomotion commands by the trajectory follower module, which tries to keep the system on the provided trajectory.

All behaviours in the plan execution need the current pose of the system in order to operate. The important part is that they need the pose of the system in the map that was used to perform the planning. For this reason it is not suitable to use the localisation output of the mapping module, since the localisation is

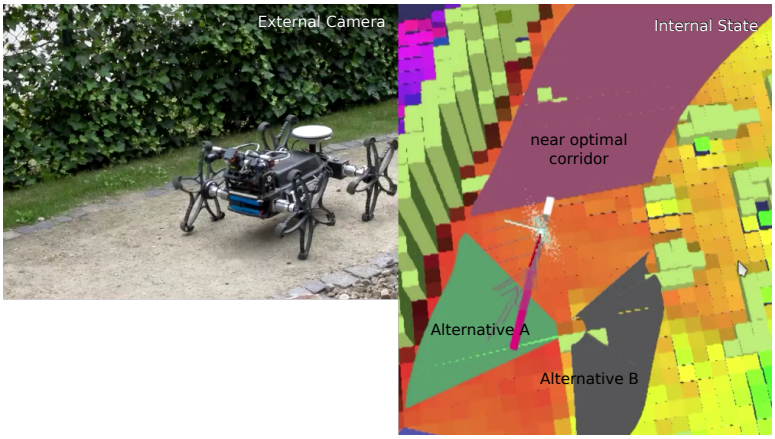


Figure 5.2: The corridor following behaviour uses the near optimal corridor generated by the path planner to direct the robot. The behaviour tries to keep the uncertainty ellipse of the system within the boundary of the corridor. The work on near optimal path planning and corridor following was performed by Sylvain Joyeux.

given for the current map, which may be very different to the one used for the planning. For this reason a second instance of the eSLAM module is running as a localisation filter. This module does not generate a map, but performs localisation on a known map like given in Section 3.1. The known map is the one which was generated when the planning loop was last triggered. The localisation filter may use different sensorial input in order to provide an estimate of the position of the system within the provided map. This pose estimate is then used in the plan execution subsystem.

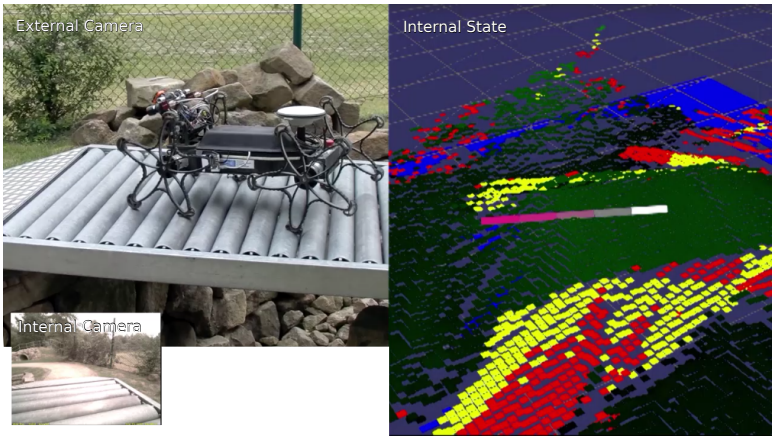


Figure 5.3: The corridor servoing behaviour uses a local representation of the surrounding to perform motion planning towards a direction which is given by the path planner. Green areas are traversable, red are non-traversable. Yellow represents shadowed areas. The work on the corridor servoing was performed by Janosch Machowinski.

5.3 Environment representation and exchange

One key technical aspect in the implementation of the navigation solution is the environment representation library *envire*. The design rationales for envire have been to provide a library which can encode environment representations and their relationships in different modalities. The advantage of using a dedicated library to this task over single representation libraries like for example Octomap is that all data which is required for the processing in the localisation module and also for the processing in the planning loop is accessible through a unified interface and all relations between the individual maps can be encoded. Further, envire provides a system to describe functional relationships between individual maps through operator interconnections. Envire also has an interface to the Vizkit visualization library in Rock, which aids debugging. All the 3D visualizations in this document have been performed by the Vizkit plug-ins for envire.

The manner how the environment is encoded in envire is based on these principle structures:

Environment Item All elements of envire have the same basic concepts. This concerns the memory handling, serialization and metadata system. The

serialization and de-serialization is split into property and map data content. Serialization is possible to a file structure for debugging as well as communication over module boundaries in the navigation stack.

Frames Frames encode spatial relationships between map elements in the Cartesian space. Frames store a transformation from the frame they encode to another frame, which they reference. The transformation can also include uncertainties. The representation and processing of these uncertainties is based on Pennec and Thirion [1997]. Each frame can have multiple Cartesian maps associated. These maps provide their information in the reference frame identified by the frame structure. The hierarchical nature of the proposed method can be encoded well in this structure. Each of the submaps has its own Frame item. The global optimisation is then applied to the individual frame transformations.

Frame Tree The Frames of an environment are structured in a tree. At the root of the tree is the world reference frame. The tree structure ensures that there is always a unique transformation between any frames of the Frame tree.

Layer All environment information is stored in a layer. There are multiple types of layers, each to encode different environment type in different ways. One such sub category are Cartesian maps. In these maps spatial information is given in the Cartesian space. This includes for example the environment representation from Section 2.1, the point-clouds from the sensor system, the feature clouds from the stereo processing pipeline, or the traversability map from the path planning loop.

Layer Tree Layers can be connected to each other in a hierarchical manner. For example multiple grid based maps could form submaps of a larger map. Another example are the map hypothesis of the particle filter. Each particle has its own map. The maps are implemented as optional children of a map hypothesis structure. The advantage of having a dedicated substructure is an optimized organisation of the data, which provides easier access to the

elements and allows operations like map processing and visualisation to be performed in a generic way.

Operators Envire not only stores the map information and their spatial relationships, but also the functional relationships. Maps can be connected by operators. These operators are often used in processing chains. For example one such chain would be the point-cloud map from the sensor is projected onto the MLS grid of the local particle. Another example is the processing chain from the MLS grid to the elevation grid to the traversability grid. The advantage of storing these chains in dedicated structures is that they can be individually triggered, and on demand processing is in principle possible.

Operator Graph The operators are connected to the maps in an operator graph. Each operator can have multiple map inputs and multiple map outputs.

Environment This is the basic structure which holds all other elements. It includes the Frame, Layer and Operator structures. A single Frame Node is referenced as the world frame of the environment.

The illustration in Figure 5.4 shows the envire graph structure which is generated in the mapping module. The sensor data is used as input to the maps and processing modules, while the structure does not change. This allows a more flexible handling for the system and easier adaptation for different sensors and system configuration.

The envire library as well as the rest of the navigation stack is available as open source from the Rock website[1]. All the navigation relevant modules use the library for internal processing of environment relevant data as well as interchange between the modules.

[1] http://www.rock-robotics.org

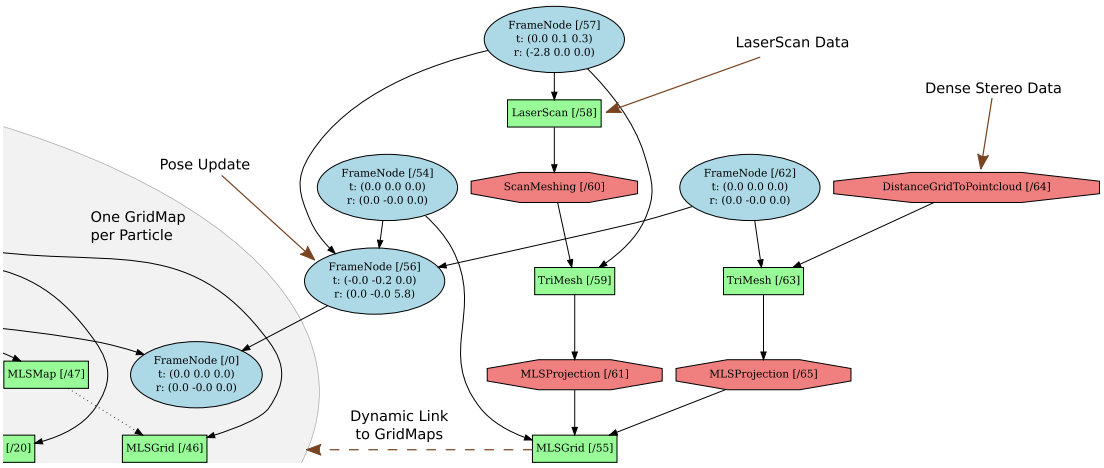


Figure 5.4: This shows the graph structure which is generated using the *envire* library to hold the relations between maps (green), transformation frames (blue) and operators (red) in the mapping module.

Chapter 6

Experimental Results

In this section a brief introduction is given into the systems which have been used for the evaluation, as well as the experiments that have been performed, and the results of those experiments.

The core hypothesis of this work is the notion of information related to the physical body of the robot. Therefore it is imperative to evaluate the proposed method on physical systems. Because of the nature of their body-environment interaction, ground based systems are in the focus for the experimental evaluation. Ground based systems – in contrast to for example airborne or underwater – have a strong interaction with their environment, since they use the ground reactive force as a means to accelerate their body mass.

The selection of the contact based odometry model in Section 2.2 already narrowed the type of ground based vehicle for the experimental evaluation. This contact based model lends itself naturally to systems which use discreet contact points with the environment for the locomotion. This is the case for walking systems – regardless the number of actual legs – as well as for leg-wheel hybrids. Wheeled systems do not immediately fit into this model. But even in the current formulation of the models, wheels can be approximated with the same method used for the Asguard leg-wheel, but with a larger number of candidate contact points.

Three different measurement models have been proposed: The shape based contact model (Section 2.3), the slope matching model (Section 2.4) and the terrain classification based model (Section 2.5). Walking systems and leg-wheel

hybrids fit directly for all three models, and have the ability to handle complex and difficult terrain. In order to evaluate the different measurement models, physical robots have been selected that fit into this class.

Experiments on the physical systems require a large number of resources. They have to be built, if no adequate models are available. They usually require a large amount of maintenance, and experiments need additional equipment for recording the environment or other experimental parameters. But even so, the physical body of the system in the real environment is central here. Using simulated environments can give clues on possible directions, but can not substitute the real systems for actual validation of the hypothesis. The experiments on the real systems form the corner stones, which ground the method in the real. Nevertheless, simulated environments have also been used in order to characterise the different aspects of the methods, especially where real experiments are not feasible or too costly in terms of resources.

6.1 Experimental Platforms

Both the odometry model (Sec. 2.2) and the visual-embodied association models (Sec. 2.3 and 2.4) rely on a discreet set of body-environment contact points $\mathcal{C}$. For this reason, systems which also have discreet contact points with the environment lend naturally to the approach. The main parts of the experiments is performed on the Asguard system, which is a leg-wheel hybrid. The method for local map building using contact based visual-embodied correspondence and the odometry was also verified on the SpaceClimber system.

6.1.1 Asguard

The Asguard v3 system (see Figure 6.1 and Figure 2.6) has a mass of approximately 14 kg and has a variable distance of 15 cm to 19 cm from the ground to the wheel axis. The twist joint freely rotates the rear axis around the main lateral axis of the system, thus ensuring that the Asguard is in contact with the environment with all four wheels at any time. For sensing the environment, the system uses position encoders on the wheels and the body twist joint. An Xsens MTi AHRS provides

Figure 6.1: The Asguard v3 (Joyeux et al. [2011]) platform has a similar mechanical design compared to the Asguard v2 (Eich et al. [2008]), but includes additional sensors and on-board processing. The main sensors used for this work are the wheel and joint encoders and an AHRS, as well as a scanning laser range finder (LRF).

orientation with respect to the gravity vector to an accuracy of around 1°. The system uses a 2D laser scanner (Hokuyo UTM-30LX). The scanner is mounted on a tilt unit, which can actively rotate the sensor circa 90°. Some of the experiments used a continuous wave motion on this joint, and in other experiments the sensor was fixed at at a 30° forward facing angle (see Fig. 6.2). There a two cameras with a 25 cm baseline, each with a resolution of 768 x 480 pixels. The robot is equipped with a Core2 Duo running at 1.5 GHz and 2 GB of memory.

Let C be the configuration space of the robot, with $c \in C = (\gamma_1, \ldots, \gamma_4, \beta)$ consisting of the four wheel angles as well as the angular position of the body twist joint. For the wheel and foot indices $i \in I_w$ and $j \in I_f$, with $I_w = 1 \ldots 4$ and $I_f = 1 \ldots 5$, the foot positions in body frame B are provided by a simple forward kinematic function:

$$p : I_w \times I_f \times C \rightarrow \mathbb{R}^3, \tag{6.1}$$

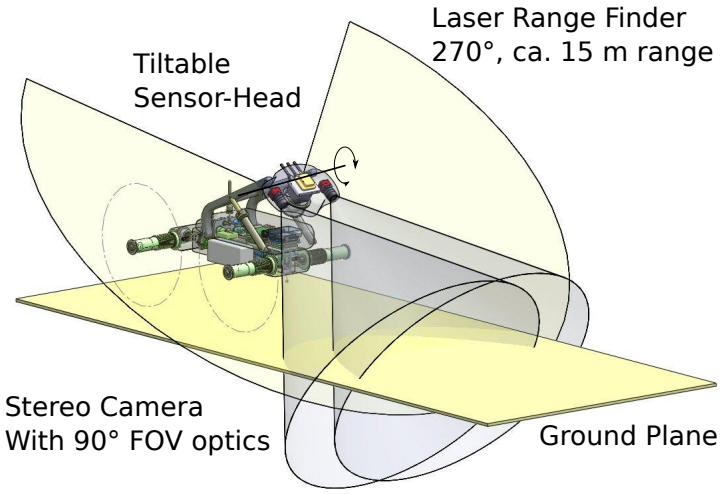


Figure 6.2: Sensor configuration of the Asguard v3 and v4 system. The main sensors used are an AHRS, joint and wheel encoders, a stereo camera system as well as a laser range finder. (CAD drawing by Felix Grimminger)

By defining a the following helper functions

$$\psi_t(i) = \begin{cases} 0 & i \text{ is front wheel} \\ 1 & i \text{ is back wheel} \end{cases} \tag{6.2}$$

$$\psi_l(i) = \begin{cases} -1 & i \text{ left wheel} \\ 1 & i \text{ right wheel} \end{cases} \tag{6.3}$$

we can easily compute the endpoints of the wheel in body frame B, using a

$$l(i,j) = \psi_l(i) \begin{bmatrix} w/2 \\ 0 \\ 0 \end{bmatrix} R_x(\theta_i + j\frac{2\pi}{5}) \begin{bmatrix} 0 \\ 0 \\ r \end{bmatrix} \tag{6.4}$$

$$p(i,j) = \psi_t(i) \begin{bmatrix} 0 \\ -l \\ 0 \end{bmatrix} + R_y(\psi_t(i)\theta_t) l(i,j) \tag{6.5}$$

where l is the wheel base and w the track width of the robot.

The set of contact points $\mathcal{C}$ is then given as

$$\mathcal{C} = \{p(w, f) | w \in I_w, f \in I_f\} \tag{6.6}$$

The Asguard system does not have sensors to measure the environment contact for the individual feet on the wheel. For that reason, the contact point estimation method described in Section 2.3.1 was used for all experiments with the physical Asguard system.

The slip-based terrain classification method requires that the ground reaction force of the contact point can be estimated. For this purpose the mechanical properties of the Asguard motor module where used. Each module contains an elastic coupling between the wheel and the gearbox. This coupling behaves like a spring. The deflection of the spring can be measured since there are encoders in front of the gearbox and on the actual wheel interface. Using a hysteresis model for the compression of the elastic coupling, the deflection of the outer versus the inner encoder can be used to estimate the torque on the wheel. This torque is further calculated into the ground reaction force required for the slip-based terrain classification based on a quasi-static model of the Asguard system.

The Asguard v3 system, which was used for most of the experiments in this thesis, was designed and built by the iMoby team.

6.1.2 SpaceClimber

The SpaceClimber (Bartsch et al. [2012]) is a six-legged walking robot. Each leg has four degrees of freedom. There is an additional joint between the front segment of the body and the rear segment. The system has an overall mass of 26 kg, which includes the sensor configuration, on-board embedded-pc and batteries. The structure of the system is manufactured using aluminum milling parts, and has a high stiffness. This is also true for the motor modules, which use brushless DC motors and harmonic drives, to minimize gear play. The maximum torque the modules can continuously provide is 54 Nm, which lead to system payloads of up to 30 kg. The SpaceClimber is equipped with a URG-04LX laser range finder and a monocular RGB CMOS camera. Both sensors are located in the head, which has an additional degree of freedom to pan left and right. In this way the LRF,

Figure 6.3: The SpaceClimber (Bartsch et al. [2012]) system is a six-legged walking robot with 25 degrees of freedom. The system has a rigid body structure and uses behavior based compliance control to adapt to the surrounding environment.

which is mounted such that the scanning plane is vertical, can be used to generate 3D scans in front of the system. For the experiments the SpaceClimber was also equipped with an XSens MTi AHRS. The feet of the system are fitted with a number of pressure sensors for which the readings are fused internally to generate an estimate of ground contact. The sensor information as well es the position data from the motor modules was logged on the systems internal 1.2 GHz Atom embedded PC, and later processed off-line. Similar to the Asguard system, a forward kinematic model was used to estimate the set of contact points $\mathcal{C}$ in body coordinates based on the motor positions. Unlike the Asguard, the SpaceClimber already provides contact information in the feet, and does not need to perform any form of contact point estimation.

6.1.3 Software tools

Asguard and SpaceClimber have a different underlying hardware and electronics basis. The low-level control in both systems is handled using either Microcontrollers (Asguard) or FPGAs (SpaceClimber). This has an impact on the software running on the systems as shown in Figure 5.1. There is a system specific software part,

which consists of hardware drivers and system specific motion modules. All high level processing however is the same software with adapted parametrization for the specific system.

All high level software processing uses the Rock framework[1]. This is a component framework, which uses a model based approach for formulating the individual runtime-components. These components are then managed at runtime in such a manner that the model constraints are fulfilled with the concrete model implementations and data connections at hand. Rock also provides a lot of tooling for managing the software modules, the connections between them, latencies for data streams as well as resolving the spatial relationships between the reference frames that are used for the data processing. The eSLAM algorithm is implemented as a library, which is encapsuled in a rock module for interfacing with other parts of the system.

The eSLAM module uses the envire library (Section 5.3) – also developed as part of this thesis – for the internal representation of the environment. The graph optimization for optimising the relationships between the individual map segments is performed using the Hog-Man[2] framework. SURF feature extraction is done using the OpenCV image processing library, and dense stereo processing was performed using *libelas* (Geiger et al. [2010]).

6.1.4 Experimental Tools

The eSLAM method does not require an absolute position reference for operation. However, in order to evaluate the performance of the method for outdoor settings, it is helpful to have position readings for reference. For this purpose the Asguard system has a Magelan MB500 RTK GPS on-board, which uses a correction signal to drastically improve the position estimation of the GPS solution. The correction signal is provided by a mobile GPS base station, which is initialized before the experiments and provided the information that its GPS antenna stays in a fixed position over the course of the experiments. The base station transmits correction information which is then used to find a GPS solution on the Asguard. This

[1] http://www.rock-robotics.org
[2] http://openslam.org/hog-man

results in position readings with an estimated accuracy of around 2 cm. Note that this accuracy was only available on the open field, and unfortunately not in the vicinity of the DFKI building due to limited visibility of the satellites.

Some of the experiments required either an a-priori map of the environment, or use such an environment model for evaluation of the mapping results. For this purpose the HDS6100 Laser Scanner from Leica Geosystems was used. For the environment models multiple scans where taken from a tripod. These scans would cover a ground diameter of around 20 m. Multiple of these scans are later registered into a complete map. The scan registration would generally result in error residuals of around 3 cm. The maps generated with this method are very dense and accurate and provide a good reference which can be assumed accurate up to 5 cm resolution.

6.1.5 Simulation Environment

While a large number of the experiments has been performed on the physical systems, some of the experiments were conducted in a simulation environment, mainly to control the environment parameters. The MARS (Römmerman et al. [2009]) simulation environment was used for this purpose. The rover which was used in the simulation is modelled after the Asguard v3 system (Section 6.1.1). The simulation provides true position and orientation values of the rover as well as information about contact points with the environment. The position values from the simulation are only used as a reference. The true orientation values are perturbed by additional noise from a simulated gyro, in order to provide realistic error characteristics for the AHRS readings. The model is a simplified version of Farrenkopf's gyro model (Farrenkopf [1978]). For the purpose of the simulation a single gyro was modelled, only taking angle random walk (rw) and rate random walk (rrw) into account, as they are considered the dominant stochastic errors. The basic model is provided in Schwendner and Hidalgo [2012].

$$\tilde{\omega}(t) = \beta_\omega(t) + n_{\text{rw}}(t) \tag{6.7}$$

$$\dot{\beta}_\omega(t) = n_{\text{rrw}}(t) \tag{6.8}$$

where $\tilde{\omega}(t)$ is the sensor error, $\beta_\omega(t)$ is the bias and $n_{\text{rw}}(t)$ and $n_{\text{rrw}}(t)$ are random variables with independent zero-mean Gaussian white noise characteristics, defined by

$$E[n_{\text{rw}}(t)n_{\text{rw}}(\tau)] = N^2\delta(t-\tau) \tag{6.9}$$
$$E[n_{\text{rrw}}(t)n_{\text{rrw}}(\tau)] = K^2\delta(t-\tau) \tag{6.10}$$

where E denotes expectation, $\delta(t-\tau)$ is the Dirac delta function. The parameters for the gyro model have been chosen to be $\text{K} = 1.5 \cdot 10^{-4} rad/s/\sqrt{s}$ and $\text{N} = 5.7 \cdot 10^{-4} rad/\sqrt{s}$. These parameters have been extracted experimentally (Schwendner and Hidalgo [2012]) for the commercially available XSens-MTI AHRS, using the Allan variance (El-Sheimy et al. [2008]) technique. The XSens-MTI is the AHRS which is also used on the physical robots.

The gyro rate error is evaluated for a single gyro only. The rate error is integrated to produce an angular error, which is then applied as a rotation around the z-axis to the orientation values from the simulation. This setup produces simulated AHRS readings for which the error is constant on the roll and pitch axis, but will drift on the yaw axis. A similar characteristic can be found on AHRS without heading compensation.

In addition to this orientation reading, the set of contact points $\mathcal{C}$ is also passed to both the odometry model and the visual-embodied measurement model. The measurement model also holds a reference to a model of the environment, which is derived from the model that was used for the simulation but may contain added noise.

6.1.6 Experiments Overview

An overview of the experiments that have been performed, as well as the relevant boundary conditions is given in Table 6.1.

Experiment Set	System	Sensors	Association Method	Terrain
Synthetic Test Data (6.2.2)	Asguard (Simulated)	AHRS, Joint Angles, Foot Contact	Embodied Localisation	Crater, Perlin Elevation, Regular Pattern, Perlin Terrain Type
Sand-field (6.2.1)	Asguard	AHRS, Joint Angles	Embodied Localisation	Open sand-field with small vegetation
Lunar DEM (6.2.3)	Asguard (Simulated)	AHRS, Joint Angles, Foot Contact	Embodied Localisation	Lunar surface
Z-Height estimation (3.2.2)	Asguard (Simulated)	AHRS, Joint Angles, LRF	Local eSLAM	Flat surface
Test track mapping (6.3.1)	Asguard	AHRS, Joint Angles, LRF	Local eSLAM	DFKI Test Track
SpaceClimber mapping (6.3.1)	SpaceClimber	AHRS, Joint Angles, LRF, Foot Contact	Local eSLAM	Flat surface with two obstacles
Terrain Classification (6.3.2)	Asguard	AHRS, Joint Angles, Stereo Camera	Local eSLAM	DFKI Test Track
Test-track loop closing (6.4.1)	Asguard	AHRS, Joint Angles, Stereo Camera, LRF	Global eSLAM	DFKI Test-track and parking lot
Sand-field online (6.4.2)	Asguard	AHRS, Joint Angles, Stereo Camera	Global eSLAM	Open sand-field with small vegetation

Table 6.1: Overview of experiments

6.2 Localisation in a-priori maps

The first set of experiments was conducted to evaluate the feasibility and characteristics of the localisation method (Sec. 3.1) in combination with the odometry model (Sec. 2.2) and the visual-embodied correspondence measurement model (Sec. 2.3). The rational of the experiments is to identify the characteristics of the method and evaluate it in practical applications. Firstly, the method is grounded using an experiment using the physical Asguard system on an open sand field. In a second set of experiments which are conducted in the simulation environment, the behaviour of the method is analysed for variations of the environment, model characteristics and filter parametrization.

For both experiments, the map of the environment is known. In addition the localisation filter receives a guess on the initial position, the set of contact point candidates and an orientation from the AHRS. The GPS is only used for reference. No exteroceptive sensors are used for the localisation.

6.2.1 Asguard sand-field

Five different runs were recorded, with travelled distances ranging from 90 m to 140 m. All runs were performed on a sand-field (Figure 6.4), with a 1 m variation in elevation. The plain areas, which constitute most of the field consist of a wave like pattern with a variation in elevation of around 5 cm, and 2.5 m peak-to-peak distance. Three of the runs are laps around a sand field track. On this track there are two obstacles with a height of 35 cm and a maximum slope of 23°. Another run loops six times around a spot which has a height variation of 0.8 m, and a single 35 cm hill as a landscape feature. On the last run, the robot moves randomly across an area of the field. The maximum obstacle size is also 35 cm. A scan of the area (see Figure 6.5) was used as an a-priori map.

For the experiments, the system was manually driven across the sand-field, and the sensor logs, including the GPS reference position recorded in log files. The logs where then processed off-line to obtain a pose estimate of the system. The position part of the pose was compared to the recorded GPS position. The runtime of the algorithm for 250 particles was 30 s for a 180 s log file on a 2.8 GHz processor.

Figure 6.4: Experimental environment for the embodied localisation experiments on the physical Asguard system. The area on which the experiments have been conducted is 30 m by 40 m in size.

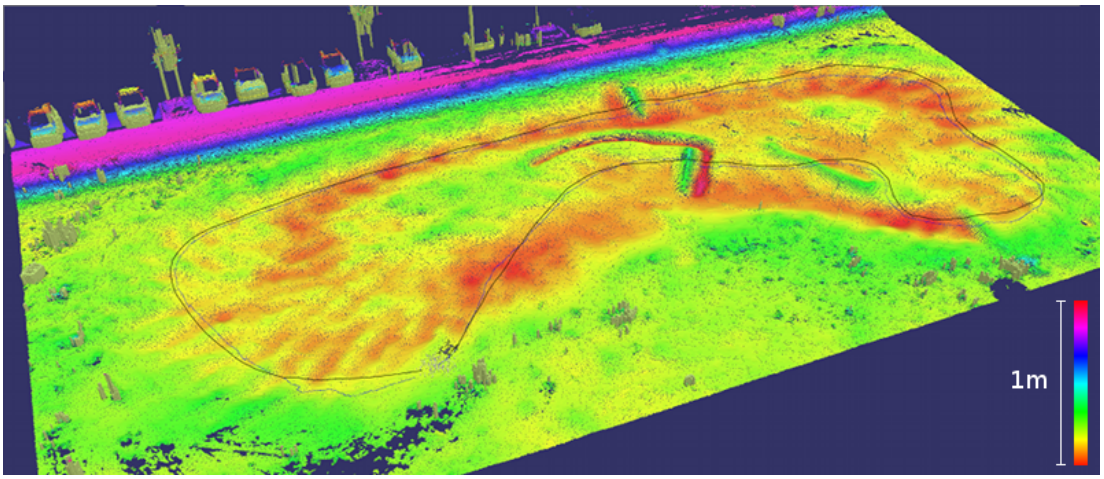


Figure 6.5: A scan of the environment was performed using a commercial laser scanner system. Seven scans where combined off-line, with matching error residuals of less than 5 cm. The resulting pointcloud was transformed into an MLS, and used as a model for the localisation experiments. The plot cycles the color range every 1 m of elevation.

Figure 6.6 and Figure 6.7 show xy position plots for one of the laps and the side loop run. The GPS reference was in RTK fixed mode, and reported an estimated error of less than 5 cm during all the runs. Since the experiments were conducted on an open field, few problems with multipath or occlusions of satellites would be expected. Thus the GPS readings can be considered very close to ground truth. The position estimate in the plots is the centroid of the particles (see (3.8)) of the position filter.

We can observe on these experimental runs that the centroid of the particles is able to track the position of the GPS within a fixed error bound, while the odometry error grows unbounded. It can also be noted that the trajectory for the centroid is less smooth compared to that of the odometry. The reason for that is likely that the terrain does not always provide sufficient distinction, which lead to a spreading of the particles. When terrain features lead to convergence of the particles this eliminates particles in other regions and generates non-continuities in the pose mean of the particles. This effect can be smoothed in retrospect, by only using the mean of particles that actually survive for a certain time period. The inherent latency required is however not acceptable for on-line use, and the problem of non-smooth best estimates of the pose needs to be handled on a higher level in the navigation subsystem.

Table 6.2 shows the results for the individual runs. The position error is the euclidean distance between the position estimate and the reference position (GPS). The mean position error is the arithmetic mean of all position errors for a single run. Position samples are taken every second. This mean gives the average distance between the estimate and the reference position for the entire run. The maximum error is the largest error distance of the entire run. One can observe that the filter reliably keeps the mean error within 0.5 m of the reference trajectory.

Further, the total distances travelled for the runs are given. The distance travelled metric accumulates the relative position changes of the estimates and the reference. This is effectively the length of the travelled path. It is notable that the odometry is consistently larger than the other two, which is most likely due to slip. The results show that the proposed method is able to compensate for this distance error.

During the runs it is noticeable that the uncertainty distribution is wider on

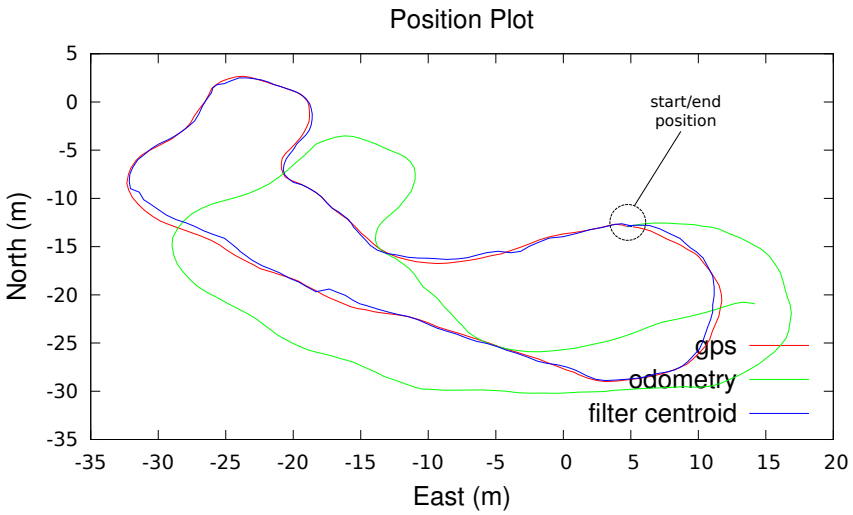


Figure 6.6: Comparing xy-position for the particle filter centroid, the odometry and the GPS for a run of 125 m around the sand field. Maximum deviation of the centroid compared to the GPS is 0.83 m for this run.

the flat parts of the terrain, and becomes narrowed down in parts by prominent terrain features. What is also noticeable that prominent terrain features act in an inverted manner. Going on flat terrain and passing a stone eliminates the particles which estimate the system on the stone.

Once the filter was adapted to the system configuration, all the runs were evaluated using the same set of parameters. Variations in the number of particles around this working point have been performed in order to evaluate the stability of the filter. The results on Figure 6.8 show near constant performance when the number of particles is above 100. This is consistent with the simulation results in Section 6.2.2.

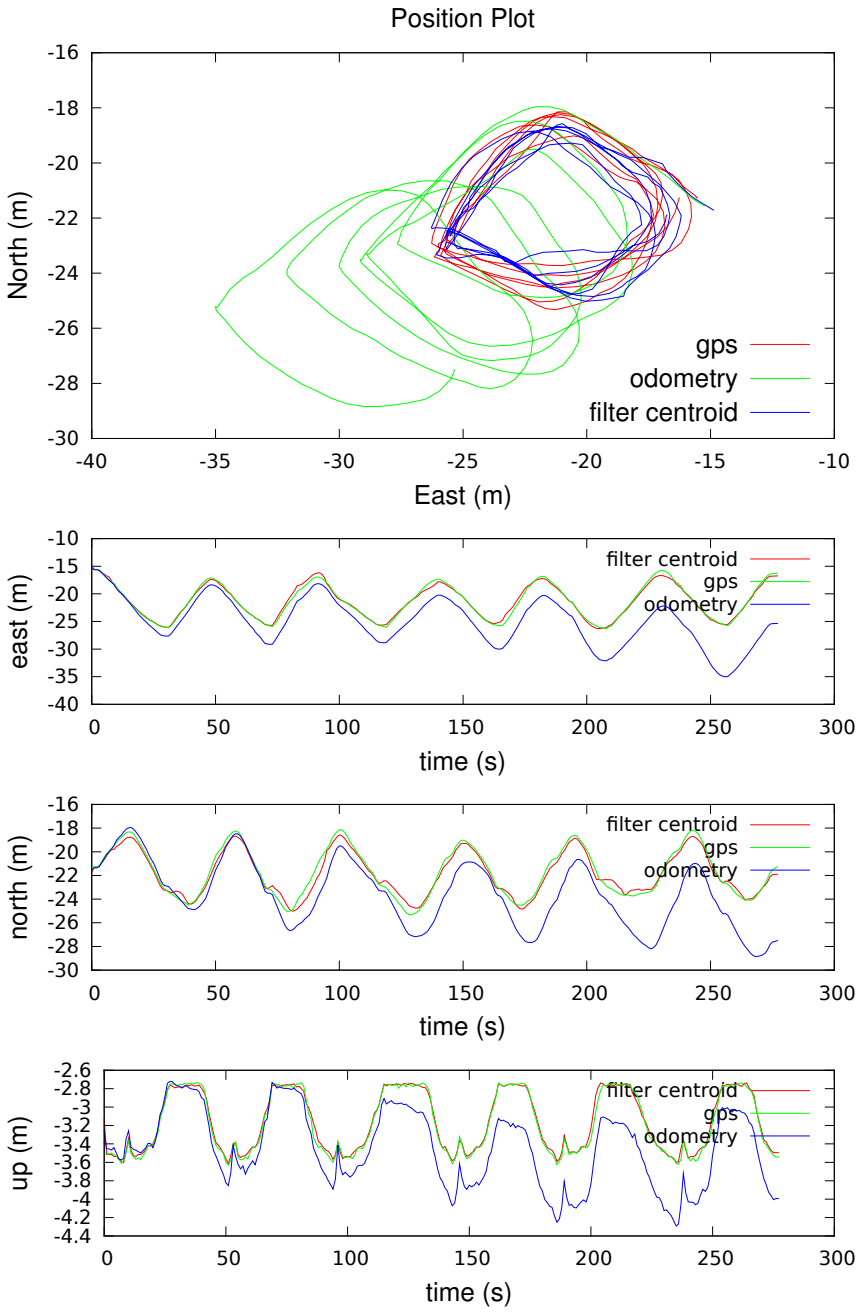


Figure 6.7: This run shows multiple loops over the same area. Over time, the odometry accumulates drift due to slip and model errors, while the filter is able to track the position within fixed error bounds.

	Mean Position Error [m]		Max Error [m]	
Run	Centroid	Odometry	Centroid	Odometry
Lap1	0.35	8.74	0.83	12.60
Lap2	0.37	9.34	1.06	12.92
Lap3	0.36	10.33	1.02	16.79
Side Loop	0.49	4.29	1.46	11.09
Cross	0.40	3.23	0.97	5.78

	Distance Travelled [m]		
Run	Centroid	Odometry	GPS
Lap1	125.83	141.97	125.19
Lap2	128.28	140.96	127.51
Lap3	124.81	135.85	123.85
Side Loop	136.84	161.63	143.89
Cross	89.67	100.31	88.46

Table 6.2: Comparing Filter Centroid and Odometry vs GPS data

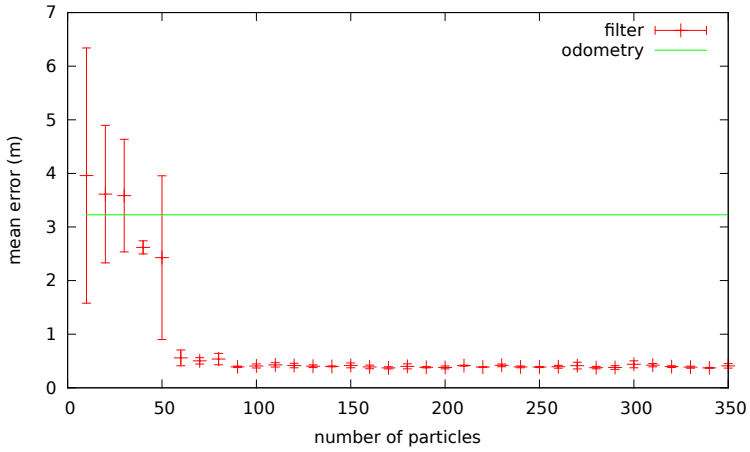
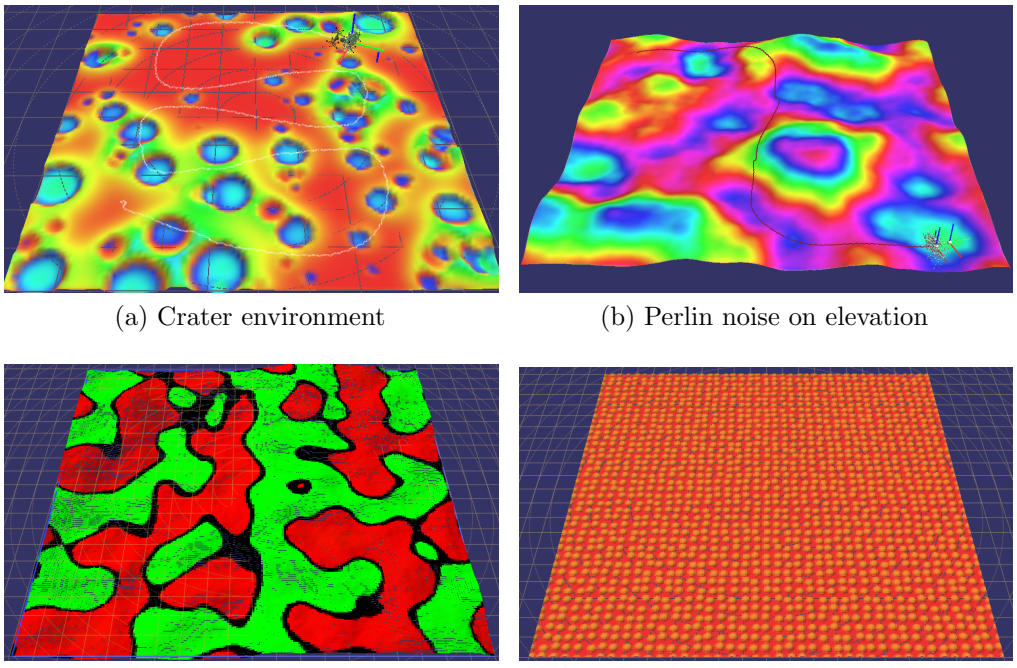


Figure 6.8: Parameter variation for the number of particles used vs mean error compared to GPS. The cross run was evaluated three times for each parameter, with a different random seed.

6.2.2 Synthetic Test Data

This set of experiments consists of synthetic terrain models which are evaluated in the simulation environment using the Asguard system model. The terrain models are generated through a deterministic random process, with an initial random seed. Multiple terrain characteristics are generated to test and compare the different measurement models.



(a) Crater environment

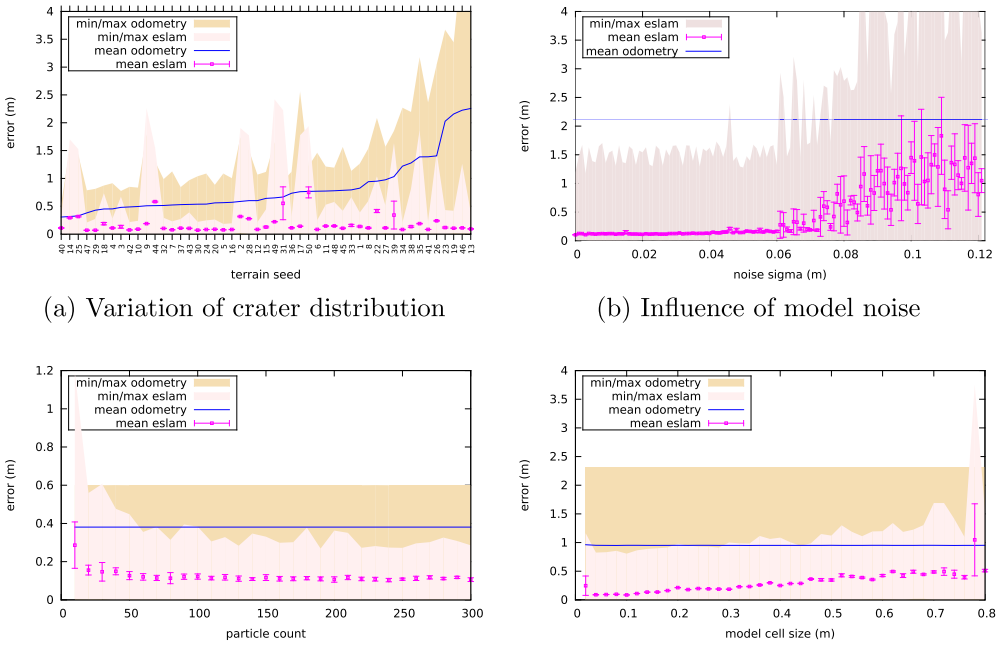
(b) Perlin noise on elevation

(c) Perlin noise on terrain type

(d) Boulder grid pattern

Figure 6.9: The performance of the localisation filter is evaluated in a simulation environment with synthetically generated landscapes. Four different types of terrain were used to evaluate different aspects of the three measurement models.

In the first set of experiments, a variations of a flat 10 m by 10 m surface, with randomly added craters (see Figure 6.9a) was used. The sizes are sampled from a uniform distribution from zero to a maximum diameter of 2 m. The depth to radius ratio is 0.375. The sampling depends on a deterministic random number generation process. Two generated terrains are the same if they have the same initial random seed. In each of the experiments the system was started at the $(-3, -3)$ position and followed a meander like trajectory of ca. 40 m distance.



(a) Variation of crater distribution

(b) Influence of model noise

(c) Variation of particle count

(d) Variation of model resolution

Figure 6.10: Terrain model and filter variations of the crater environment (see Figure 6.9a). In all plots, each parameter set was evaluated 5 times for the eSLAM filter. The plots show the mean of the mean error distance, and the 1-sigma error bars. The min/max plots are for the minimum and maximum errors over all runs.

The first experiment in the set was to identify the influence of the distribution of craters on the localisation result. For that purpose 50 randomly generated terrains with different initial seeds were evaluated. The results are shown in Figure 6.10a. Each of the terrains was simulated once and the odometry evaluated. Then, the localisation filter was run 5 times for each trajectory with different initial seeds for the particle filters random processes. The plot shows the mean error versus the reference position for the odometry and the mean and standard deviation of the individual means of the localisation runs. The graph also shows the min/max error boundaries. These are the minimum and maximum errors from all 5 evaluations. The results show that the mean eSLAM error is for most of the terrains either at around 0.1 m or for a few instances close to the odometry error. Close inspection of these runs have shown that these outliers all generated a terrain where the system was unable to progress and got stuck inside a crater, but would continue moving its wheels. Currently the error model from the odometry does not cover this case and thus the performance of the filter drops back to the level of the odometry. That the eSLAM error is bounded is not surprising, since it uses the a-priori map for global localisation, while the odometry is based on pure dead reckoning. But one particular aspect to note in this plot is that there seems to be no significant correlation between the odometry error and the eslam error.

In the second experiment, the influence of environment-model (map) noise on the filter performance is investigated. For that purpose one of the randomly generated terrains from the first experiment was used to generate the trajectory and simulation data. The model that was given to the filter is then perturbed by Gaussian noise on the cell level. Figure 6.10b shows the influence of this noise on the performance of the filter. While a noise level below 0.05 m has a small linear relation with the mean error, noise levels above this threshold generate a larger deviation in the mean error and generally decrease the performance down to the level of pure odometry. It is surprising that the filter can handle model noise with a sigma up to 0.05 m without noticable performance impact. Likely, this can be attributed to the fact that the noise is uncorrelated, and can be compensated well by the particle filter. The point at which the noise starts to impact the result likely depends on the contact measurement noise value σ^2_{meas} (2.22). Choosing this value too small results in a narrower particle distribution, which gets distracted by

noise very easily. A higher value for this parameter results in lower error correction abilities of the filter.

In another experiment, the number of particles for the representation of the pose distribution was varied on a single terrain. Figure 6.10c shows the results. It can be noted that for this particular instance a particle count of more than 150 did not improve the mean error any further, but seems to have a small influence on the maximum error.

The results in Figure 6.10d show a variation of the cell size of the model that was given to the particle filter. The graph shows an expected increase in mean error with an increasing cell size. The wheel-base of the Asguard system is approximately 0.5 m, and should have an influence on the results. If the system fits within a single cell, the contact model has no additional value to provide. It seems however that the transition between cell boundaries provide enough information to generate a smooth error transition. Why there is an outlier at the 0.02 m cell resolution is not clear, and might be connected to the simulation environment.

One very interesting aspect of the shape based model is the requirement to have a structured environment. This is necessary in order to differentiate different positions and adapt the posterior accordingly. To verify this property, an experiment was conducted where a fixed number of craters are scaled in height. A scale value of 0 results in a flat terrain. The previous experiments where conducted with a fixed scale value of 0.7, which roughly corresponds to the maximum elevation in metre. The simulation is run on the crater terrain. The localisation filter receives a model of that terrain with added Gaussian noise with a standard deviation of 0.1 m. The resulting elevation maps for different scales are shown in Figure 6.11.

Like in the other experiments, the localisation filter was run 5 times per scaled terrain and the mean and max errors compared to odometry. The resulting plot in Figure 6.12 supports the intuition that the shape based measurement model is only useful when the terrain has enough shape features. The performance of the localisation filter is roughly on par with pure odometry for scale values less than 0.1. It improves until 0.25 and then seems to plateau. At scale values of 0.8 and larger some of the craters are too steep for the simulated Asguard to traverse and it will get stuck. The simulation is stopped if the position has not changed for

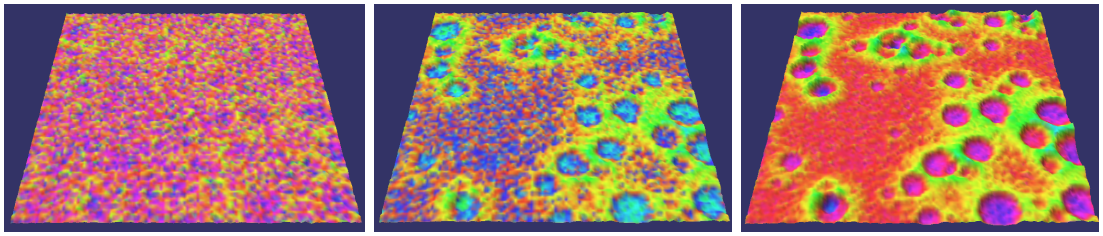


Figure 6.11: Terrain models with a common crater distribution but different scale values (left: 0.1, center: 0.5, right: 1.0). The terrain model has an added background noise with sigma of 0.1. This makes the terrain shape for the scale value of 0.1 barely distinguishable from the noise.

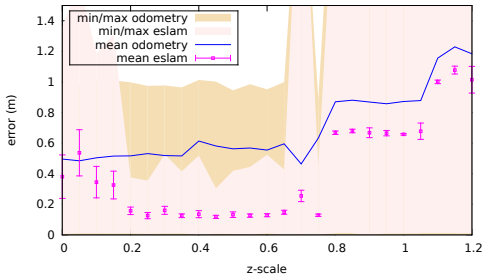
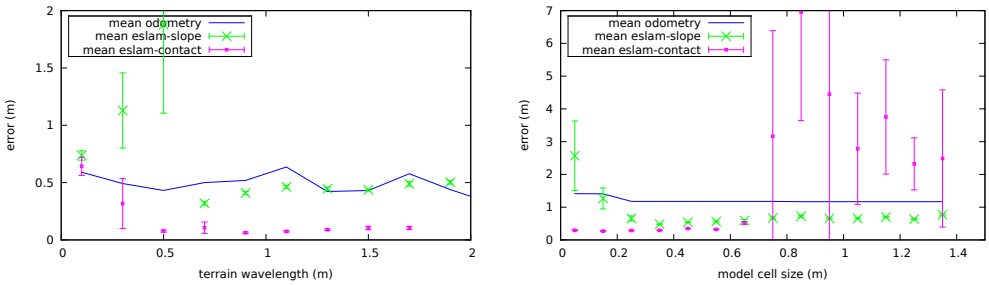


Figure 6.12: Localisation filter performance results for different scale values of the same crater distribution. Low scale values result in equal performance for localisation filter and odometry. The scale values from 0.8 upwards lead to shorter overall trajectories as the craters become too big to be traversed.



(a) Variation of pattern wavelength (b) Variation of model resolution

Figure 6.13: Comparison of the contact and the slope based measurement models on perlin noise pattern.

5 s. The localisation filter is still better in these situations, but the relative error difference is smaller due to the shorter overall trajectory.

In a second range of experiments, the influence of the spatial frequencies has been investigated, and how the model resolution influences the relationship between the contact based and the slope based measurement model. For this purpose, a different type of synthetic terrain based on Perlin noise was used (see Figure 6.9b). Perlin noise Perlin [2002] has the characteristics of producing irregular patterns that are in a specific range on the spatial frequency spectrum. This type of noise is particularly suited for the generation of artificial terrains.

The plot in Figure 6.13b shows the two terrain shape based measurement models with a variation of the model resolution. Like in Figure 6.10d a decreasing performance for the contact based model can be noted. Once the resolution is bigger than 0.7 m, the filter becomes unstable and performs significantly worse than odometry. The slope based model shows an inverted behaviour. This model performs poorly at high resolution, but increases performance at a resolution of around 0.5 m at which it becomes stable. These results reflect the design characteristics of the models. The contact based model performs much better at high resolution data, and is thus also better qualified to be used in a SLAM setting for the short ranged data associations. The slope based model is more stable at lower model resolutions, and could for example be used to perform global associations. The crossing point for the performance of the two models is roughly the size of the robot.

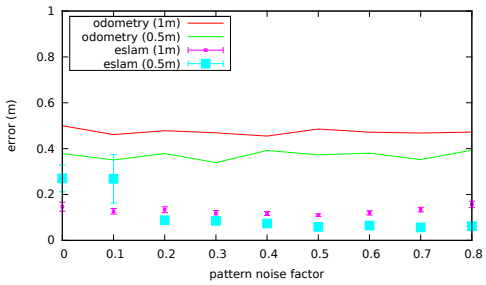
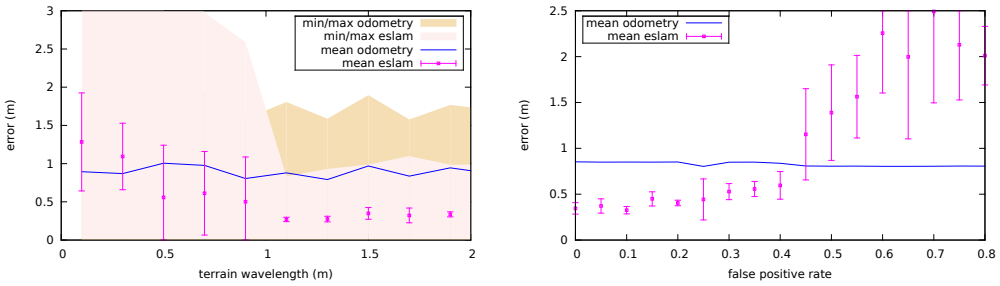


Figure 6.14: Localisation results for a grid pattern of same sized bumps with grid spacing of 1 m and 0.5 m on a 20 m by 20 m flat terrain. A pattern noise factor of 0 represents a completely regular grid, while a factor of 1 is completely random.

Another interesting aspect is the influence of the terrain wavelength. The plot in Figure 6.13a shows the results for a terrain, where the maximum slope was held fixed by scaling the height value by the wavelength. The zero point would in this case represent a completely flat terrain. As expected, the contact based model requires a certain amount of structure to perform, and has its maximum performance around 0.5 m. The terrain model for the slope based measurement was held fixed at 0.5 m for this experiment. Any frequency components below this value are ignored and lead to bad performance. The filter even performs worse with higher wavelength up to the model resolution threshold. After this threshold, the slope model keeps at the same performance. Note that here this is roughly the same value as the odometry. This is due to the low size of the map. Validation of larger maps like in Figure 6.13b shows that the filter error is bounded, while the odometry error is not. Again, this is due to the fact that the filter in these experiments has the a-priori map as a global reference.

So far, the terrains used for the experiments used irregular patterns. The question is how the regularity of repeating patterns influences the results. For this purpose grid pattern of half-spheres with a diameter of 0.2 m was added to a 20 m by 20 m flat terrain. Each grid position was perturbed by a scaled uniform noise in x and y direction. A scale factor of 0 resulted in a completely regular pattern, while a scale factor of 1 was completely irregular. The plot in Figure 6.14 shows that the regularity does not have a significant influence on the result if the pattern has a 1 m grid spacing. For a grid spacing of 0.5 m however, high regularity of the pattern results in lower performance. This is likely attributed to

particles jumping between grid cells. This effect can appear once the grid spacing is smaller than the spread of the uncertainty distribution of the filter.



(a) Variation of wavelength

(b) Classification false positive rate

Figure 6.15: Evaluation results for the terrain classification based measurement model.

Finally, the terrain classification based measurement model was also evaluated in the simulation environment. The terrain for the experiments has a perlin noise elevation profile on a 20 m by 20 m grid. The map contains two different terrain classes, as can be seen in Figure 6.9c. The plot in Figure 6.15a shows that a pattern wavelength of below 1 m does not provide stable results. This is in contrast to the results for the contact based model in Figure 6.13a. The difference is likely attributed to the fact that the signal is discrete, and the update frequency much lower in the terrain classification case. The plot in Figure 6.15b shows the results for a pattern with a fixed wave length of 2.5 m, and a variation of the actual false positive classification rate. The parameter $p_{\text{correlated}}$ was set to 0.6. This seems to also provide the lower boundary for increasing the performance. Although the results are not as stable due to the low update frequency of the characterisation signal, false positive rates as high as 0.4 still provide a gain in localisation performance.

6.2.3 Lunar DEM

In Section 6.2.2 the influence of the cell size on the particle filter performance was evaluated. It is clear that the shape based measurement model (Section 2.3) is limited to a cell resolution close to the size of the robot. One practical scenario

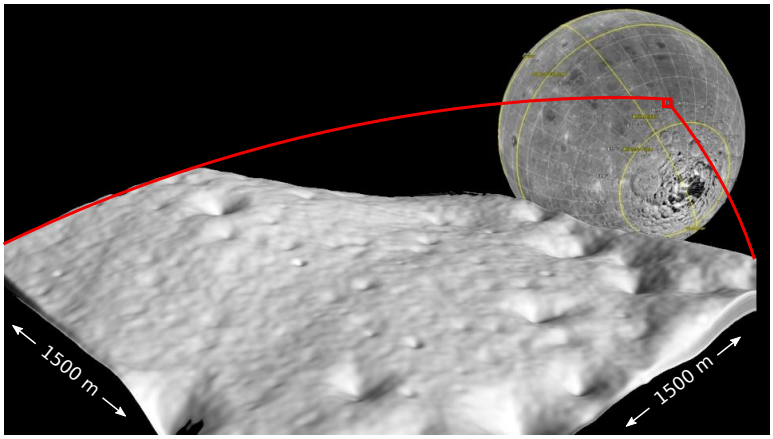


Figure 6.16: A rendered image of the DEM which was used for the experiments. The height values are not to scale. The difference between the lowest and the highest point for this 1500 m by 1500 m patch is 118 m. The patch was extracted from a DEM (LRO_NAC_DEM_60S200E_150cmp.tif), which was made publicly available by the United States Geological Survey (USGS) through the LMMP website.

for the application of the visual-embodied correspondence is localisation on a planetary surface. GPS is not available in these environments, and there is no magnetic field. The Lunar Reconnaissance Orbiter (LRO) generated digital elevation maps (DEM) of up to 1.5 m resolution Moratto et al. [2010].

A DEM of 1500 m by 1500 m from the Aitken Basin near the lunar south pole extracted from publicly available data (see Figure 6.16). Like in Section 6.2.2 the MARS simulator was used to generate position and orientation values as well as environment contact points for a trajectory on this DEM with the Asguard v3 model. The total simulation time for the trajectory is 70 min. The total distance the rover has travelled is 2300 m, which results in an average velocity of 0.54 $^{m}/_{s}$. The loop the rover follows, starts and ends at the same position. The overall height difference between the highest and the lowest point of the trajectory is 32 m.

The initial condition of the particle filter is a sampling around the start position of the trajectory, which is considered known. The orientation with respect to the map is considered unknown and needs to be recovered by the filter. The t_{meas} (see Section 2.4) variable was set to the cell size of the DEM.

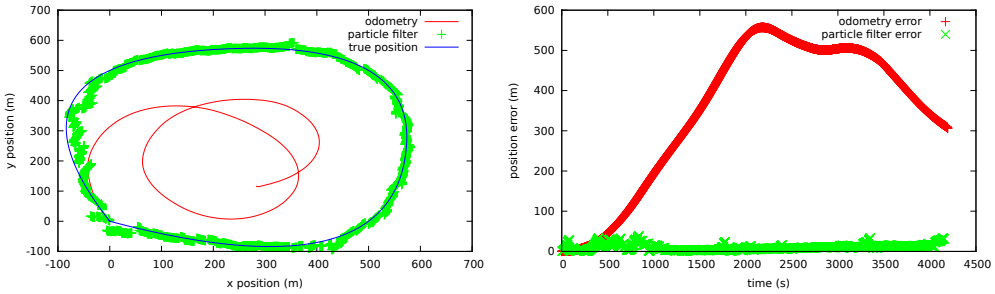


Figure 6.17: The result of the localisation experiment are shown on the left graph. The true position in x and y-direction is plotted in blue. The odometry (red) exhibits significant deviation from the true position, mainly caused by the gyro noise which results in heading errors. The estimate of the particle filter (green) is able to track the position and exhibits a bounded error behavior. The graph on the right shows that this error is bounded to 40 m while the odometry grows up to 500 m

The orientation values are perturbed by gyro noise. The integration of the rate noise results in a final heading error, as reported by the AHRS, of approximately 340°. The gyro noise parameters were chosen to reflect characteristics of low-precision MEMS gyroscopes, and thus represent a very basic sensor configuration. The trajectory the odometry follows can be seen in comparison to the reference and the filter result in Figure 6.17. The error in heading has a significant impact on the trajectory of the odometry. The global position estimate deviates strongly from the reference trajectory, and is only meaningful locally. In contrast, the particle filter, which is run using 500 particles is able to compensate this error. The error profile for the odometry is much smoother, but grows up to an error of 500 m, while the filter error stays within a 40 m boundary. The error is calculated as the length of the distance vector between the position on the true pose vs the position on the estimate at any particular time step. The minimum and maximum values for the filter error are 0.9 m and 38.8 m, with an average of 11.7 m. The experiments were performed on an Core-i7 processor with 2.8 GHz clock frequency. The runtime for the filter was around 10 min for the 70 min simulation time log. The code was not optimized for this particular task and could be made significantly faster.

6.3 Local Map generation

In Section 6.2 the localisation part of the described method has been evaluated in simulation and on a physical system. No exteroceptive sensors are used in this context, but a priori-maps are available. For scenarios where this is not the case, a map needs to be generated in a SLAM setting. In this section, a number of experiments are conducted on the real system to validate the local map generation method described in Section 3.2. The measurement models used here are the shape based model as well as the terrain characterisation model.

6.3.1 Shape based mapping

In order to evaluate the generation of local maps using the visual to embodied data correspondence, a metal bridge was traversed on the DFKI test-track. During the run, the wheels had a significant amount of slip. Using the contact model from Section 2.3 in the mapping process, this slip was compensated, as can be seen in the map generated by the particle with the highest weight in Figure 6.18. The height profile of the trajectory was compared against the reference map from the commercial laser scanner and the map generated by pure odometry in Figure 6.19. While the odometry fails to track the terrain profile and generates a height error of approximately 0.6 m, the SLAM with visual-embodied correspondence stays within 5 cm accuracy. The local map generation algorithm requires ca. 30% CPU load on a 2.8 GHz Core-i7 for processing of 100 particles at a travel speed of 0.5 $^{m}/_{s}$, odometry processing at 100 Hz and map updates every 2 cm.

A second experiment was performed to qualitatively evaluate the effectiveness of the approach also on other type of systems. Here, the SpaceClimber (Section 6.1.2) was used in a test scenario, where the system traversed a rock of around 20 cm height placed on flat ground. Figure 6.20 shows a sequence of the internal camera, the filter state and an external camera at different time points. It can be observed that the rock in the centre of the scene is only visible at time 3 s. Once the system is actually on the rock, the external camera as well as the laser scanner have no way of getting information on the relation of the rock to the system, and in a classical set-up would have to infer its position using external references. Using the proposed method the slip between 31 s and 33 s is detected and corrected for.

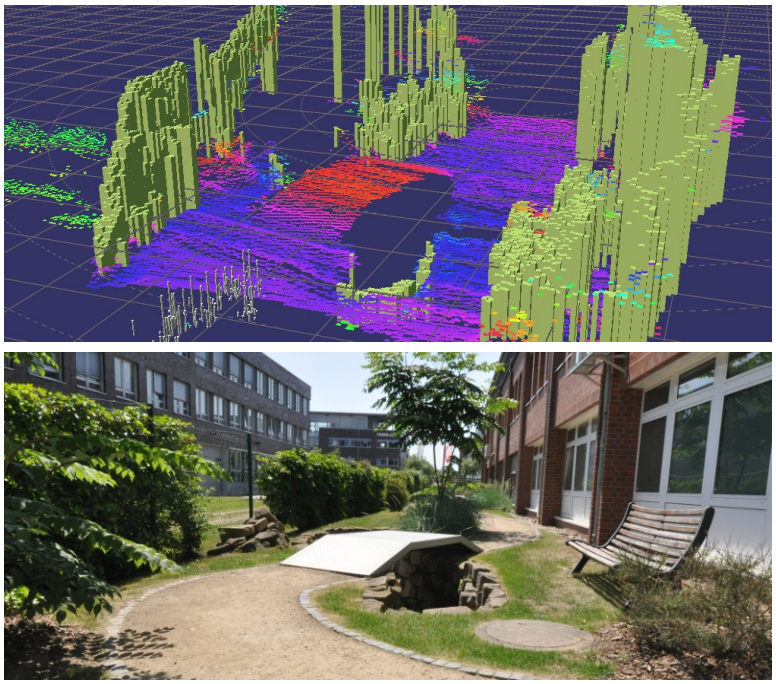


Figure 6.18: In this experiment the Asguard v3 system was crossing a bridge on the test-track (bottom), which caused significant wheel slip. The method proposed for generating local maps using embodied data is able to smooth the result and provide a locally accurate map (top) of the environment.

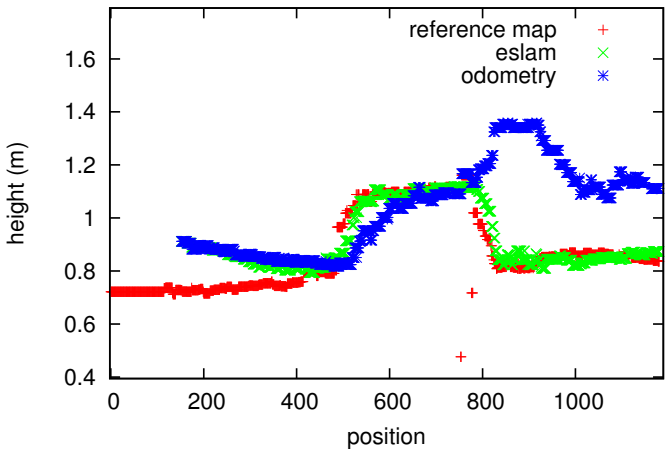


Figure 6.19: Height profile of the map, which was generated passing the test-track bridge. The results from the embodied SLAM method are compared to the reference scan and the odometry.

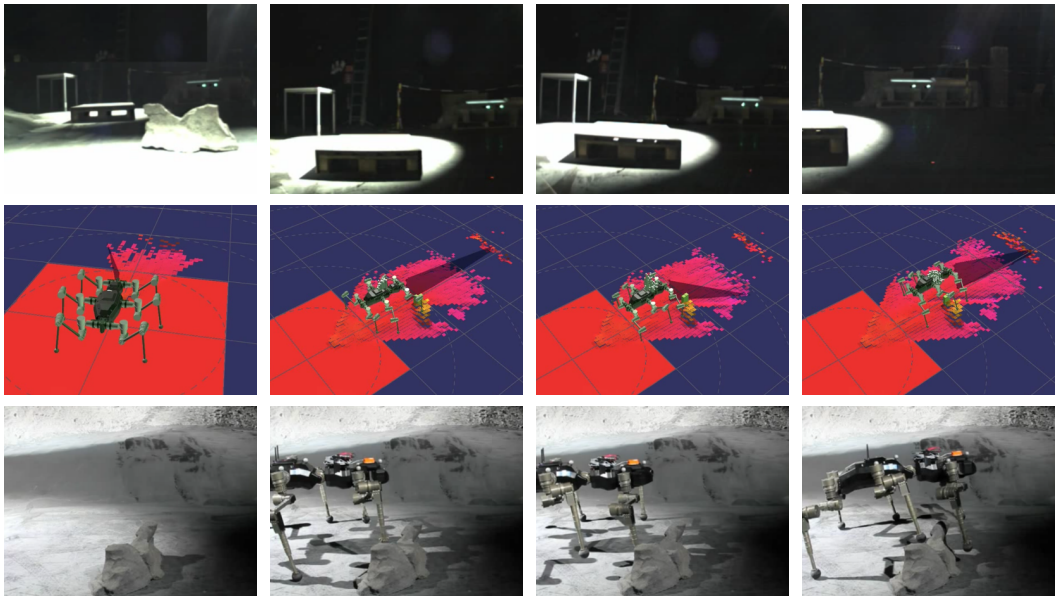


Figure 6.20: Sequence of using visual-embodied mapping on the SpaceClimber system. The top image row is the systems internal camera, the center row a visualisation of the state of the mapping filter, which shows the map of the current best particle. The bottom row is a view from an external camera. The columns are taken at time 3 s, 31 s, 33 s and 50 s.

Even when the system is well over the obstacle at 50 s, the information on contact or non-contact will correct the relative positioning and map in retrospect.

6.3.2 Terrain classification

Instead of using shape or slope based information as shown in the experimental results in Section 6.2.1 and 6.2.3, it was stated in the introduction that the indirect form of embodied data can also be used to provide useful information. In order to test this hypothesis, an experiment was conducted on the DFKI test track, where the Asguard system traversed the complete length of the test track. At a variable frequency of around 0.5 Hz stereo distance images are generated. Further, the colour image information from the left camera is classified using the visual terrain classifier described in Section 2.5.1 to colour the pixels of the stereo distance image according to the classification for path and grass. At non-regular intervals, the slip detection system will detect slips, mainly when the system is turning. The torque profile based classification method from Section 2.5.2 provides a classification for the terrain of the slipping wheel. Figure 6.21 shows a graphical representation of the internal state of the SLAM filter for these cases.

The gathered log has a length of 125 s and spans a distance of around 70 m. Figure 6.22 shows the resulting map, which matches reasonably well with the original. The visual classifier was able to classify the map on relevant parts along the trajectory of the robot.

Looking at the errors in Figure 6.23, we can see that the method described was able to improve over odometry for the test track run in most cases. In the beginning of the log, odometry is still better, but accumulates error over time. This is also the case for the filtered positions. However, the error seems to be reduced by the error models employed. The strongest improvement was visible when using a combination of the shape based and the slip based update methods.

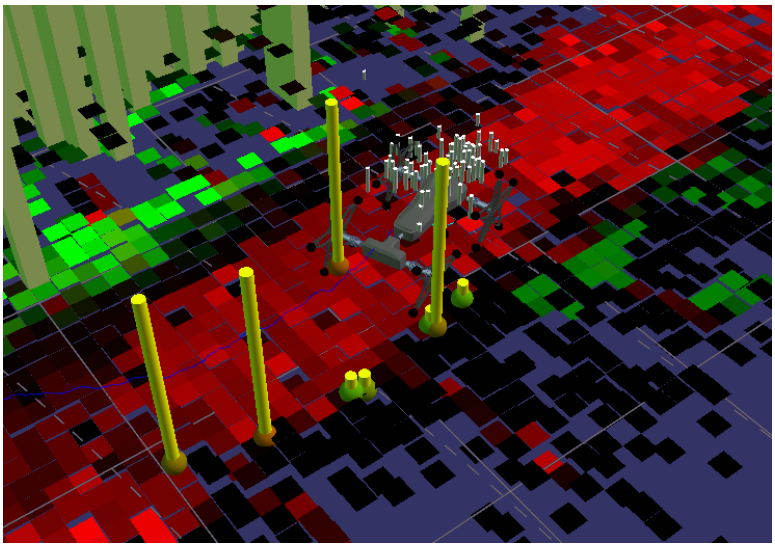


Figure 6.21: Visualisation of internal filter state for the test track run. The particles are shown as grey bars with their height proportional to the particle's weight. The yellow bars represent points where slip based classification events occurred, and how well (bar height) they matched with the classification from the visual data.

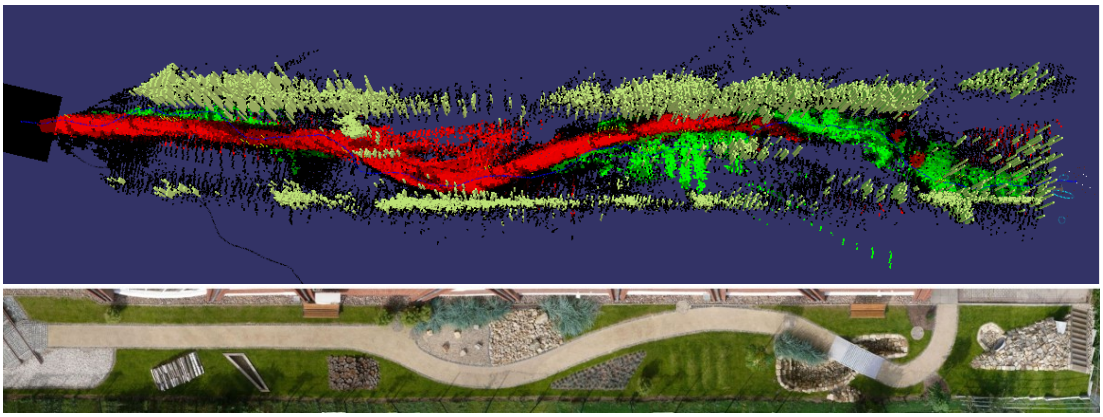


Figure 6.22: Resulting map for the test track run (top). Red and green parts are identified terrain types, while yellow indicates walls. Parts that could not be visually classified are black. (bottom) Top view of test track for reference.

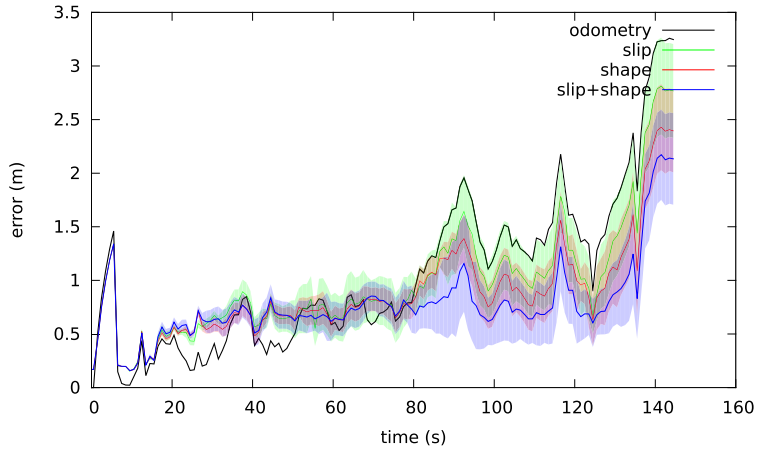


Figure 6.23: Errors compared to reference trajectory, for test track run. For each of the different measurement models, the filter was run 5 times using different random seeds. The mean and error bars are plotted over time. Both error models improve over odometry and even more so when used together.

6.4 Global Map Generation

In this final part of the experiments section, we focus on the generation of globally consistent maps by combining locally generated map segments using the method proposed in Section 4. All experiments here have been performed using the shape based visual-embodied association method. The association of map segments is done using the natural sequencing of segments through the odometry as well as the visual association method described in Section 4.1.

6.4.1 Test Track Loop closing

In this experiment the Asguard v3 system traversed a loop around the DFKI building. The local map segments are generated using the laser range finder and the shape based visual-embodied update method. Each segment has an associated stereo image which is used for matching map segments according to the visual matching method. The dense stereo image is not used for the mapping as in this experiment it generated too many erroneous map readings. Once the building is traversed, a loop closure is identified by the visual matching mechanism, and the segment poses are corrected. The maps in Figure 6.24 show a top down

view before and after loop closing. The visualization shows the individual map segments and the robot path for each individual particle with reference to the local frame. The segments are connected such that each new segment starts at the mean of the final positions of the previous segments particles. After loop closing, the segments relative position is updated according to the constraint graph. The final map is generated based on the maps of those particles that generate the smoothest path. Smooth in this context is when the trajectory does not have discontinuities between two map segments. Before the correction by the pose graph optimizer, the segments are connected such that the new segment starts at the centroid of the final position of the previous segment. Figure 6.25 shows the transition between two map segments after correction.

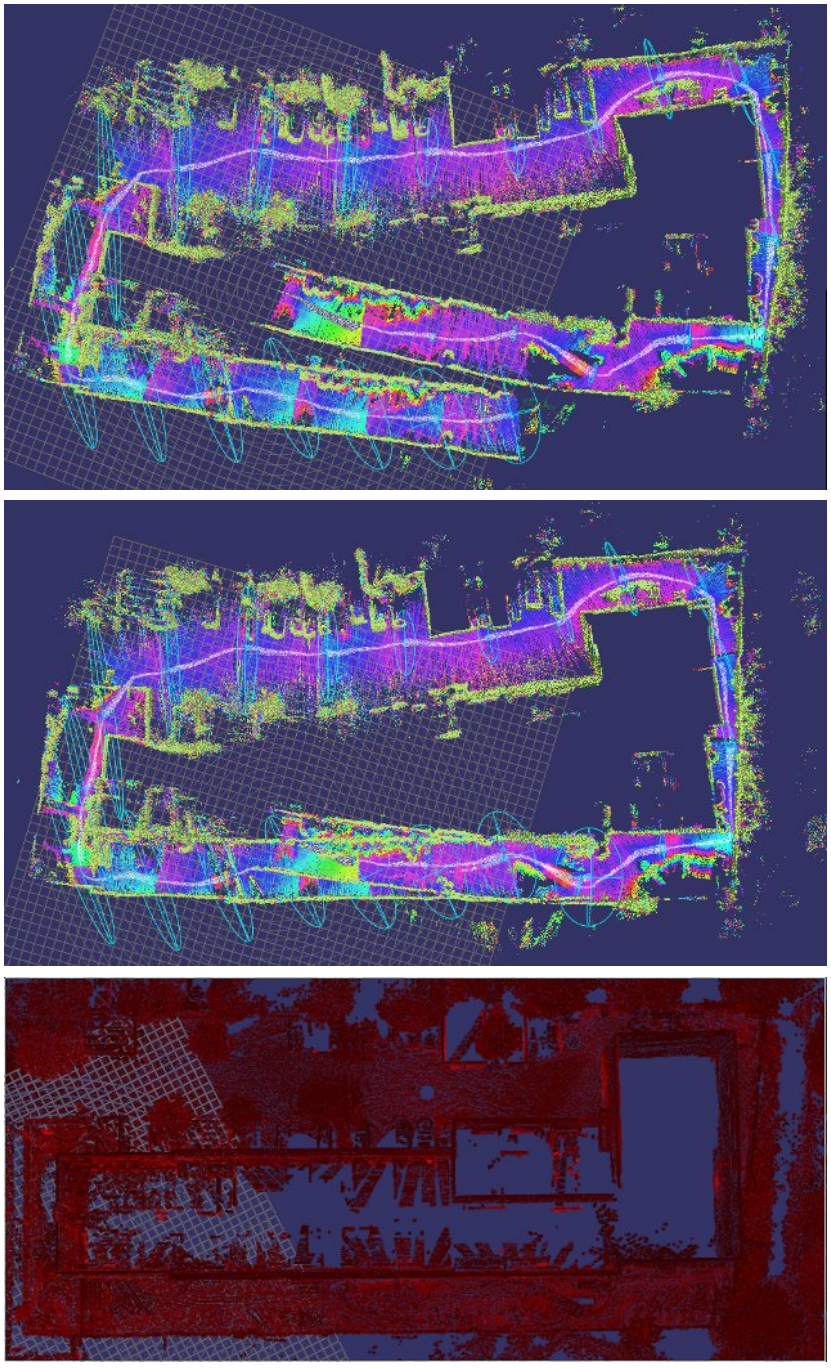


Figure 6.24: Loop around the DFKI building before (top) and after (center) loop closing. Loop closing is performed by matching stereo image features. When compared to a scan of the building (bottom) shows that the optimized map is locally consistent, but contains distortions.

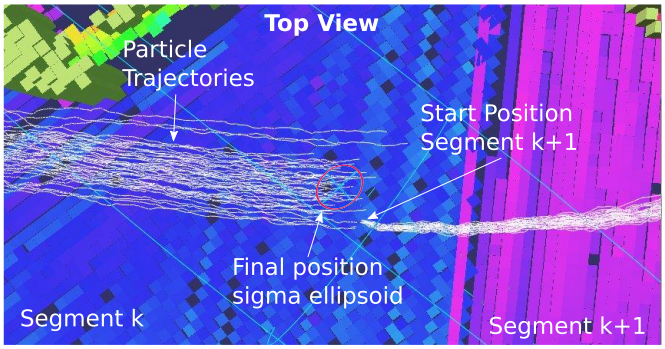


Figure 6.25: Enlarged top view of a transition between two segments of the globally optimized map. The particle trajectories of the previous segment (left) are connected to the trajectories on the following segment (right). Map parts are selected from the individual segments after optimization in such a way that the robot path is kept smooth (compare Figure 4.5). This ensures smooth map transitions and globally matching maps.

6.4.2 Sand-field on-line

In a final experiment the applicability of the proposed method for on-line navigation in real environments is tested. A sand-field with uneven ground and small vegetation was used as the testing ground, as can be seen in Figure 6.26. The goal of the experiment was to perform fully autonomous navigation in an unknown environment. For that purpose the robot was placed on the field, an given a command to traverse to a location 35 m away. The architecture of the navigation stack and the software setup is described in Section 5.2. All calculations including the map generation was performed on the 1.5 GHz Core2 Duo embedded PC on the system. The travel speed was 0.2 m/s. After the system has reached the target point, it was commanded to go back to the origin using the newly acquired map of the surrounding.

Figure 6.27 shows a top view of the environment map at two different stages. An overlay plots the current path plan in to the map. A more detailed view of a map segment is given in Figure 6.28. Here the system is just in front of some vegetation, and needs to perform replanning.

After the target position is reached, a plan to return to the origin is generated and followed using the localisation filter. Figure 6.29 shows the internal state of the filter. The localisation filter also uses the shape based visual-embodied

Figure 6.26: Asguard v3 system on sand-field which was used to perform the experiments.

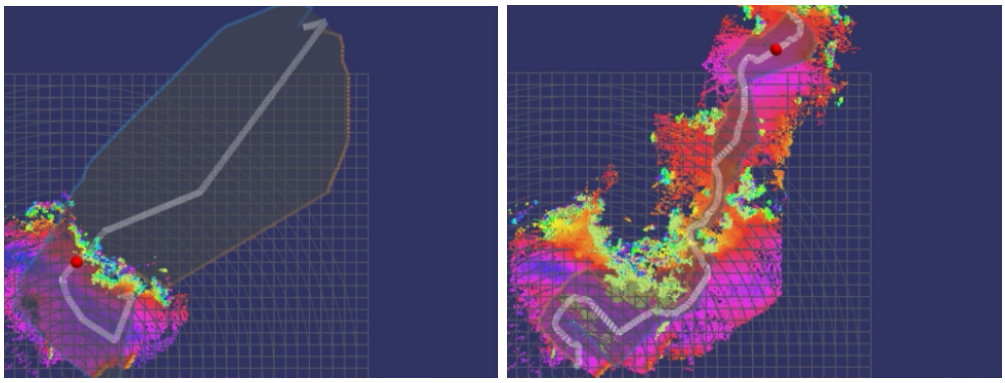


Figure 6.27: On-line map generated on the autonomous run over the sand field. (left) Shows the map at an early stage, and includes the current near optimal corridor for navigation. (right) The final map after approx. 50 m of travel.

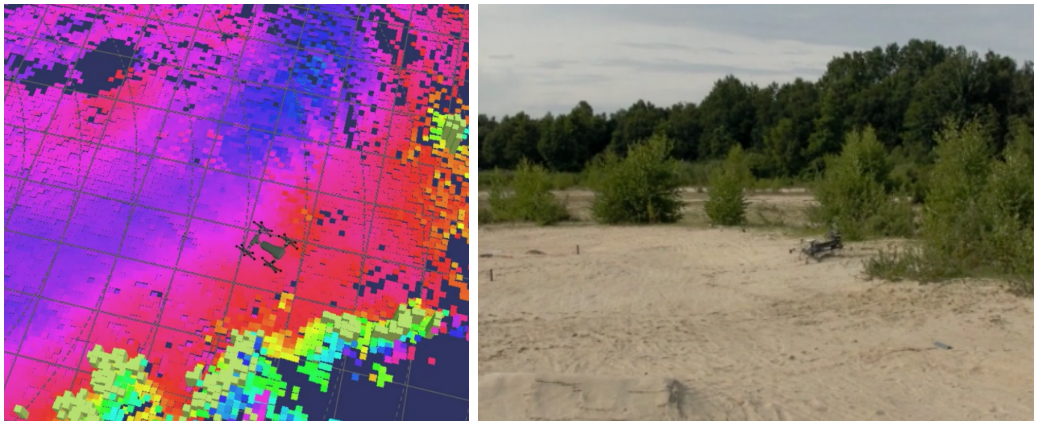


Figure 6.28: Detailed view of the generated map in front of a vegetation obstacle. (left) shows the current best particle of the on-line generated maps, and (right) a picture from an external camera.

correspondence with the elevation map as an environment model. One can observe that for passages with little terrain features, the particle distribution spreads wide, while more distinct parts of the trajectory make th filter converge to a smaller area.

The mapping process was performed using 30 particles while the localisation used 150 particles. Both navigation filters combined required about 90% CPU load of one of the processor cores on average.

For this experiment only the local map building method was used, since the global smoothing requires constraints to be generated between the segments. In the given experimental set-up however, the stereo vision based constraint generation is not well suited, since the places revisited are only visible from the opposite viewing direction, and no previously seen visual features can be identified.

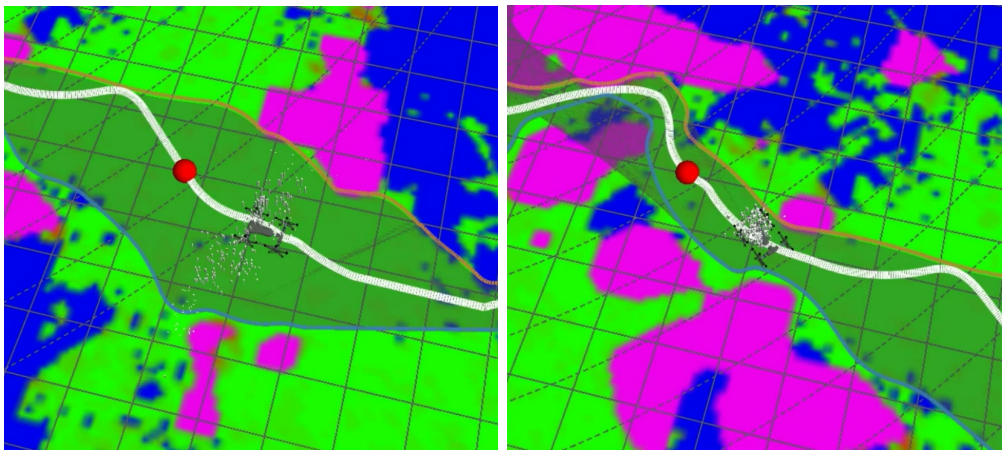


Figure 6.29: After the system arrived at the target position, the autonomous navigation system switched into localisation mode. A traversability map is generated from the acquired map. Green parts are traversable, and purple parts of the map are considered obstacles. A plan is generated back to the start position. Shown is the dark near optimal corridor, and the corridor center as a white line. The grey bars represent particles of the localisation filter on a wide (left) and narrow (right) passage.

Chapter 7

Conclusions

7.1 Thesis Summary

There is little doubt that we are only at the beginning of a technological development which will see mobile intelligent machines embedded into our everyday life. However, the ease that we take for granted at handling the complex environment that we inhabit does not easily translate into simple mechanisms which can take reliable sensor data and generate actuator commands based on fixed rules. Most information that can be gained from sensors is ambiguous at best and potentially misleading in the worst case. We have to teach our systems how to live with this situation and judge the information available under the relevant context. Relying on single sensor modalities – like vision – withholds relevant information to make those judgements. There is most certainly a reason most animals including humans have developed a large array of senses to judge situations by combining different modalities. Crossing a street could be a dangerous endeavour when relying purely on vision and ignoring auditive clues or even vibrations of the ground. We should apply the same insight when making robots. Computer vision and vision based localisation and mapping has produced marvellous results in a large number of fields. They provide the basis for most of the advanced navigation capabilities that we see in mobile robots today. One should however make a distinction between the field of computer vision and that of robotics. Seeing the robot as a platform that carries around the cameras is not doing justice to the physical

presence of the system in the physical environment its operating in. A robot offers more potential for interaction and learning insights about the environment than is provided purely by optical sensors. The need for this may not have been so obvious in the past ten years, where the application of successful use of optical sensor based methods have lead to self driving cars and fully autonomous aerial vehicles. The new challenges for robotics are however in the strong interaction with difficult environments. Robots that can climb, run, swim, manipulate and all other sort of things humans are capable of are what we will be looking at in the next decade to come. One indicator for this is the current DARPA challenge, which features humanoid robots for complex mobile manipulation tasks. While computer vision will likely continue to provide a key role for robotics cognition, these systems will require more than optical sensors to perform their tasks in a safe and reliable manner.

The belief that the interaction of the physical body with the environment has a potential to provide additional information for the robot's navigation capabilities have motivated this thesis. The term embodied data has been introduced, to make a distinction between the exteroceptive sensory information provided by sensors based on optical principles like cameras or laser range finders and sensory information that is closely related to the physical interaction of the robot with its environment. The underlying mechanism of localisation and map building is that of data association. The same things are perceived from different viewpoints. Based on this relation between the things and the viewpoints, consistent spatial models of the environment can be generated. Data association is the identification of the same things. When extending the spectrum of modalities available to the robot by embodied data, this data association can be performed across modalities. The visual-embodied data association is the primary concept used in this thesis. The robot sees something and later on perceives it with its body. The tools of Bayesian filtering are used to process this data association in a way such that spatially consistent localisation estimates and maps can be generated.

Different measurement models have been proposed that capture different aspects of the environment and the interaction of the environment with the system. The terrain shape based model matches potential body-environment contact points with the contour of the terrain. The slope based model can be

used when the resolution of the terrain shape is lower than the size of the robot. The terrain class model can be applied on flat terrain, which is distinguishable only by how the robot interacts with the terrain.

The goal of the thesis was to generate 3D environment models of complex terrains that are suitable for navigation. The proposed measurement models are most suited to be used with a Rao-Blackwellized Particle Filter. The problem with this type of filter though is that of particle depletion for longer runs. To overcome this problem, a hierarchical SLAM approach was presented, which generates local map segments using the RBPF, and joins these segments together using global constraints. The segment's relations to each other are optimized using a GraphSLAM based approach. The resulting errors are back propagated into the map segments, and the map segments merged into a final 3D map representation of the environment, which can be used for navigation.

The algorithms have been implemented in software modules, which are then integrated into a complete navigation solution. Further, a software library was developed which allows the representation of environment models in a structured and consistent manner.

All components of the proposed solution are verified experimentally. The experiments have been conducted in simulation, and on two physical systems, Asguard and SpaceClimber. Part of the software modules were integrated into a complete autonomous navigation stack, and verified on the Asguard system. The Asguard was able to travel 35 m in complex and unknown terrain, and return to the origin without human operator interaction. This supports the original goal that the approach should generate maps suitable for navigation.

7.2 Lessons Learned

The central subject of this thesis is the application of embodied data for localisation and mapping in the context of navigation for mobile exploration robots. The data association between different modalities plays a pivotal role in this context. Four different data association scenarios have been offered (see Figure 1.12) in this thesis. The *odometry* is a classical method for dead-reckoning navigation, which is a baseline for the other methods. When an a-priori map is available, the *embodied*

Update Model	Requirements	Costs	Benefit
Contact Model	Terrain with features size greater two times model noise. Model resolution less than system size.	Cell lookups per 5 cm travel = particles x contact points	Bounded position error better than system size
Slope Model	Terrain with wavelength component greater system size. Model resolution greater than system size.	Cell lookups per 0.5 m travel = particles	Bounded error worse than cell size
Classification Model	Body measurement differentiable terrain type with wavelength greater two times vehicle size. False positive rate less than 0.4.	ca. Cell lookups per 0.5 m travel = particles	Bounded position error better than system size

Table 7.1: Comparison of measurement models

localisation approach can be used to provide a bounding of the global error. Three measurement models are presented to perform the association of local embodied percepts to a map. While it is always delicate to generalise experimental data, especially in complex real world settings, an attempt is made here to extract key attributes of the individual solutions. For this purpose, the respective requirements, costs and benefits for each model are estimated in Table 7.1 based on the results of the experiments in Section 6.

All measurement models provide a means to reference a global map, and therefore bound the position error. A practical implementation was shown that was able to bound the error to a mean of 0.5 m for a path length of over 130 m. All measurement models also have a low computational complexity when compared to visual-visual association methods and could be implemented on systems with low computational resources. The run-times given in the experiments can only provide an indication, as much of it is implementation specific. How much resources the approach would take in other implementation is better indicated by the complexity and estimated update frequency given in Table 7.1. The parallel nature of the particle filter also makes the approach suitable for efficient implementation on

FPGA hardware.

The embodied localisation method does not require extra sensors which go beyond what is usually available on exploration systems (AHRS and joint encoders), and can operate without a vision system. This could make the approach interesting for low-mass or low-resource systems or missions in environments with difficult visual properties.

It was also shown how the visual-embodied data associations can be used in a *local eSLAM* setting, when no a-priori map is available. This setup has the same requirements for the terrain as the localisation case. As was shown in the experiments, it can compensate the odometry error in high-slip situations. Also it is possible to correct the map at places that are not in view of the visual sensors anymore. The corrections of the map happen in the vicinity of the body, where it is most relevant for the navigation system. Like in the localisation case, the visual-embodied data associations are cheaper to compute than visual-visual correspondences, due to the sparse nature of the data. They can of course also be used in combination with a visual SLAM system.

The ability to add explicit constraints in a *global eSLAM* setup is the another contribution of this paper. By back propagating the pose error of a pose optimisation graph into the map, it is possible to generate maps with smooth transitions in the intersection between map segments. This is a requirement for using these maps for path planning. It was shown that the *eSLAM* can be used in a practical navigation implementation for an exploration rover. The method was used to map an area of 35 m diameter, and return to the origin with a planned path based on this map.

7.3 Limitations and Future Work

As was shown in the experimental section, currently the approach is limited to systems with discreet body-environment contact points, like walking robots or wheel-leg hybrid systems. Transferring the approach to wheeled systems, with the use of the contact point estimation method shown in Sec. 2.3.1 should be straight-forward, though.

Further, the current model works under the assumption that the environment

is solid. Terrain that deforms strongly when in contact with the robot only works up to a certain tolerance. While the experiments worked well for the sand-field where sinkage was around 3 cm, traversing larger vegetation and deep grass will likely make the approach fail.

The proposed method for generating global constraints is currently based on a vision method which only matches poses that have a similar viewing direction. Returning to a place with a different orientation is likely to result in no match.

The embodied SLAM approach is a collection of environment models, prediction models and measurement models in combination with a hierarchical SLAM solution which integrates the robot-environment interaction to generate navigable maps. It is not meant as a replacement for visual methods, but rather as an opportunity to augment existing solutions. As was shown in this work, it can stand on its own, and can be used for scenarios where optical sensor based systems are not suitable. The vision however is that it expands the current possibilities to integrate body-environment interaction into navigation solutions and result in lower requirements for the vision system while improving robustness.

Modelling the possibility of deformable terrain and including it in the mapping step would likely increase the value and applicability of the approach.

Another area is to use active sensing in order to maximise the information return for the navigation system. For example, stepping on a stone is a large information gain when the terrain around the stone is flat.

Currently the way to perform loop closing is performed using visual-visual association. One could also use visual-embodied associations here to have a more consistent framework.

Even though the experiments were limited to using the locomotion subsystems of ground based robots, using mobile manipulation systems with very rich body environment interaction and dense sensor coverage are very likely to improve the ability to localise and map the environment. Especially the manipulation aspect has the ability to increase the robot's awareness in the environment because of the additional interaction points.

For the navigation of autonomous mobile robots, the robots have not actually been required in a lot of proposed approaches. I think that with the advent of increasingly more complex systems the robot body will gain in relevance also for

the navigation. The work presented here provides some insight into why this information is useful, and how it can be used to improve robot navigation.

References

Achint Aggarwal and Peter Kampmann. Tactile sensors based object recognition and 6d pose estimation. In Chun-Yi Su, Subhash Rakheja, and Honghai Liu, editors, *Intelligent Robotics and Applications*, volume 7508 of *Lecture Notes in Computer Science*, pages 406–416. Springer Berlin Heidelberg, 2012.

A. Angelova, L. Matthies, D. Helmick, and P. Perona. Fast terrain classification using variable-length representation for autonomous navigation. *Computer Vision and Pattern Recognition (CVPR)*, 2007.

Ronald C Arkin and Tucker Balch. AuRA : principles and practice in review. *Journal of Experimental & Theoretical Artificial Intelligence*, pages 37–41, 2010.

T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): part II. *IEEE Robotics & Automation Magazine*, 13(3):108–117, September 2006. ISSN 1070-9932.

Stephen Barkby, Stefan Williams, Oscar Pizarro, and Michael Jakuba. An efficient approach to bathymetric SLAM. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 219–224. Ieee, October 2009.

Sebastian Bartsch, Timo Birnschein, Malte Römmermann, Jens Hilljegerdes, Daniel Kühn, and Frank Kirchner. Development of the six-legged walking and climbing robot SpaceClimber. *Journal of Field Robotics*, 29(3):506–532, 2012. ISSN 1556-4967. doi: 10.1002/rob.21418.

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3): 346–359, 2008.

N. Bergman, L. Ljung, and F. Gustafsson. Terrain navigation using Bayesian statistics. *IEEE Control Systems Magazine*, 19(3):33–40, June 1999. ISSN 02721708. doi: 10.1109/37.768538. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=768538.

Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

J-L Blanco, J. Gonzalez-Jimenez, and J-a Fernandez-Madrigal. An Alternative to the Mahalanobis Distance for Determining Optimal Correspondences in Data Association. *IEEE Transactions on Robotics*, 28(4):980–986, August 2012.

Jose-Luis Blanco, J A Fernandez-Madrigal, and J Gonzalez. Toward a Unified Bayesian Approach to Hybrid Metric–Topological SLAM. *IEEE Transactions on Robotics*, 24(2):259–270, 2008. ISSN 15523098.

Michael Bosse, Paul Newman, John Leonard, and Seth Teller. SLAM in Large-scale Cyclic Environments using the Atlas Framework. *The International Journal of Robotics Research*, 2004.

Christopher A Brooks and Karl Iagnemma. Self-Supervised Terrain Classification for Planetary Surface Exploration Rovers. *Journal of Field Robotics*, pages 1–24, 2012. doi: 10.1002/rob.

Rodney A Brooks. New approaches to robotics. *Science*, 253(5025):1227–1232, 1991.

Wolfram Burgard, Cyrill Stachniss, Kai Arras, and Maren Bennewitz. Advanced Techniques for Mobile Robotics - SLAM Front-Ends, 2011. URL http://ais.informatik.uni-freiburg.de/teaching/ws11/robotics2/pdfs/rob2-13-frontends.pdf.

Sebastian Carreno, Philip Wilson, Pere Ridao, and Yvan Petillot. A survey on terrain based navigation for auvs. In *MTS/IEEE OCEANS*, 2010. URL http://eprints.soton.ac.uk/162213/.

S. Chitta, P. Vernaza, R. Geykhman, and D.D. Lee. Proprioceptive localization for a quadrupedal robot on known terrain. In *Proc. IEEE ICRA*, 2007.

A. Clark. *Being There: Putting Brain, Body, and World Together Again*. Bradford Books. MIT Press, 1998. ISBN 9780262531566.

A.J. Davison and N. Kita. 3D simultaneous localisation and map-building using active vision for a robot moving on undulating terrain. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–384–I–391. IEEE Comput. Soc, 2001.

Mehmet Dogar, Vishal Hemrajani, D Leeds, B Kane, and S Srinivasa. Proprioceptive Localization for Mobile Manipulators. Technical Report February, Carnegie Mellon University, Robotics Institute, 2010.

A Doucet, S Godsill, and C Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and computing*, 2000a.

Arnaud Doucet, N De Freitas, Kevin Murphy, and Stuart Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, 2000b.

H. Durrant-Whyte, D. Rye, and E. Nebot. Localization of autonomous guided vehicles. In *Robotics Research - International Symposium*, volume 7, pages 613–625. MIT Press, 1996.

Markus Eich, Felix Grimminger, Stefan Bosse, Dirk Spenneberg, and Frank Kirchner. Asguard: a hybrid legged wheel security and sar-robot using bio-inspired locomotion for rough terrain. In *IARP/EURON Workshop on Robotics for Risky Interventions and Enviromental Surveillance*, 2008.

N. El-Sheimy, H. Hou, and X. Niu. Analysis and modeling of inertial sensors using allan variance. *IEEE Transactions on Instrumentation and Measurement*, 57(1):140–149, 2008.

Alberto Elfes. Using Occupancy Grids for Mobile robot perception and navigation. *Computer*, 22:46–57, 1989.

Carlos Estrada, J Neira, and JD Tardós. Hierarchical SLAM: Real-time accurate mapping of large environments. *Robotics, IEEE Transactions on*, pages 1–9, 2005.

Nathaniel Fairfield, David Wettergreen, and George Kantor. Segmented SLAM in Three-Dimensional Environments. *Journal of Field Robotics*, 27(1):85–103, 2010.

R. L. Farrenkopf. Analytic Steady-State Accuracy Solutions for Two Common Spacecraft Attitude Estimators. *Journal of Guidance, Control, and Dynamics*, 1(4):282–284, 1978.

Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Charles Fox, Mat Evans, Martin Pearson, and Tony Prescott. Tactile SLAM with a biomimetic whiskered robot. *IEEE International Conference on Robotics and Automation*, 2012.

Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

Udo Frese. Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping. *Autonomous Robots*, 21(2):103–122, August 2006. ISSN 0929-5593.

B. Gassmann, F. Zacharias, J.M. Zollner, and R. Dillmann. Localization of Walking Robots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1471–1476. Ieee, 2005.

Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient large-scale stereo matching. In *Asian Conference on Computer Vision (ACCV)*, 2010.

Simon Godsill. Sequential Monte Carlo Methods. In *Machine Learning Summer School*, 2009.

J.P. Golden. Terrain contour matching (tercom): A cruise missile guidance aid. *Image processing for missile guidance*, 238:10–18, 1980.

N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *Radar and Signal Processing*, volume 140, 1993.

Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. IEEE ICRA*, 2005.

Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1):34–46, February 2007. ISSN 1552-3098.

Giorgio Grisetti, R Kummerle, and Cyrill Stachniss. Hierarchical optimization on manifolds for online 2D and 3D mapping. In *IEEE ICRA*, 2010.

F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. In *Proc. IEEE Transactions on Signal Processing*, 2002.

R. Haarmann, R. Jaumann, F. Claasen, M. Apfelbeck, S. Klinkner, L. Richter, J. Schwendner, M. Wolf, and P. Hofmann. Mobile Payload Element (MPE): Concept Study For A Sample Fetching Rover For The ESA Lunar Lander Mission. *Planetary and Space Science*, 74(1):283–295, 2012.

T. Hague, JA Marchant, and ND Tillett. Ground based sensing systems for autonomous agricultural vehicles. *Computers and Electronics in Agriculture*, 25 (1):11–28, 2000.

Ibrahim Halatci, Christopher A. Brooks, and Karl Iagnemma. A study of visual and tactile terrain classification and classifier fusion for planetary exploration rovers. *Robotica*, 26(6):767–779, 2008.

D. Helmick, A. Angelova, and L. Matthies. Terrain adaptive navigation for planetary rovers. *Journal of Field Robotics*, 26(4):391–410, 2009.

J. Hertzberg and F. Kirchner. Landmark-based autonomous navigation in sewerage pipes. In *Advanced Mobile Robot, 1996., Proceedings of the First Euromicro Workshop on*, pages 68–73. IEEE, 1996.

Mark a Hoepflinger, C David Remy, Marco Hutter, Luciano Spinello, and Roland Siegwart. Haptic terrain classification for legged robots. In *IEEE International Conference on Robotics and Automation*, pages 2828–2833. Ieee, May 2010. ISBN 978-1-4244-5038-1. doi: 10.1109/ROBOT.2010.5509309.

R. Hoffman and E. Krotkov. Terrain mapping for outdoor robots: robust perception for walking in the grass. In *Proc. IEEE ICRA*, 1993.

Jeroen D. Hol, Thomas B. Schön, and Fredrik Gustafsson. On resampling algorithms for particle filters. In *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, pages 79–82. IEEE, 2006.

L. Hostetler and R. Andreas. Nonlinear kalman filtering techniques for terrain-aided navigation. *Automatic Control, IEEE Transactions on*, 28(3):315–323, 1983.

K Iagnemma, S Kang, H Shibly, and S Dubowsky. Online terrain parameter estimation for wheeled mobile robots with application to planetary rovers. *Robotics*, 20(5), 2004.

Sylvain Joyeux and Frank Kirchner. Leaving choices open in planner/planner integration. In *ICAPS Workshop on Planning and Plan Execution for Real-World Systems*, 2009.

Sylvain Joyeux, Jakob Schwendner, Frank Kirchner, Ajish Babu, Felix Grimminger, Janosch Machowinski, Patrick Paranhos, and Christopher Gaudig. Intelligent Mobility - Autonomous Outdoor Robotics at the DFKI. *KI - Künstliche Intelligenz*, 2011.

Yasir Niaz Khan, Philippe Komma, and Andreas Zell. High resolution visual terrain classification for outdoor robots. In *ICCV Workshops*, pages 1014–1021. IEEE, 2011.

Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613. Ieee, May 2011.

Etienne Le Grand and Sebastian Thrun. 3-axis magnetic field mapping and fusion for indoor localization. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, pages 358–364. IEEE, 2012.

PC Lin, H Haldun Komsuoglu, and D Koditschek. Legged odometry from body pose in a hexapod robot. In M.H. Ang Khatib and O., editors, *Experimental Robotics IX*, pages 439–448. Springer, 2006.

Tim K Marks, Andrew Howard, Max Bajracharya, and Larry H Matthies. Gamma-SLAM : Visual SLAM in Unstructured Environments. *Journal of Field Robotics*, 26(1):26–51, 2009.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI Conference on Artificial Intelligence*, 2002.

Z. M. Moratto, M. J. Broxton, R. A. Beyer, M. Lundy, and K. Husmann. Ames Stereo Pipeline, NASA's Open Source Automated Stereogrammetry Software. In *Lunar and Planetary Institute Science Conference Abstracts*, volume 41 of *Lunar and Planetary Inst. Technical Report*, page 2364, March 2010.

Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. 6D SLAM 3D Mapping Outdoor Environments. *Journal of Field Robotics*, 24: 699–722, 2007.

Edwin B. Olson. *Robust and Efficient Robotic Mapping*. PhD thesis, Massachusetts Institute of Technology, 2008. URL http://hdl.handle.net/1721.1/44713.

X. Pennec and J.P. Thirion. A framework for uncertainty and validation of 3-D registration methods based on points and frames. *International Journal of Computer Vision*, 25(3):203–229, 1997. ISSN 0920-5691.

Ken Perlin. Improving noise. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 681–682. ACM, 2002.

P. Pfaff, R. Triebel, and W. Burgard. An Efficient Extension to Elevation Maps for Outdoor Terrain Mapping and Loop Closing. *The International Journal of Robotics Research*, 26(2):217–230, February 2007.

Felix Rehrmann, Jakob Schwendner, John Cornforth, Dick Durrant, Robert Lindegren, and Per Selin. A Miniaturised Space Qualified MEMS IMU for Rover Navigation Requirements and Testing of a Proof of Concept Hardware Demonstrator. In *11th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2011.

M. Römmerman, D. Kuhn, and F. Kirchner. Robot design for space missions using evolutionary computation. In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pages 2098 –2105, may 2009.

M. Rosheim. *Leonardoś Lost Robots*. Springer, 2006.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010. ISBN 978-0-13-207148-2.

Jakob Schwendner. Map Segmentation based SLAM using Embodied Data. In *IEEE International Conference on Multisensor Fusion and Information Integration (MFI)*, 2012.

Jakob Schwendner and Javier Hidalgo. Terrain aided navigation for planetary exploration missions. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS-12)*, 2012.

Jakob Schwendner and Sylvain Joyeux. Self Localisation using Embodied Data for a Hybrid Leg-Wheel Robot. In *IEEE ROBIO*, 2011.

Jakob Schwendner and Frank Kirchner. eSLAM - Self Localisation and Mapping Using Embodied Data. *KI - Künstliche Intelligenz*, 2010.

Jakob Schwendner, Felix Grimminger, Sebastian Bartsch, Thilo Kaupisch, Mehmed Yuksel, Andreas Bresser, Joel Bessekon Akpo, Michael K.-G. Seydel, Alexander

Dieterle, Steffen Schmidt, and Frank Kirchner. CESAR: A lunar crater exploration and sample return robot. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3355 – 3360, 2009.

Jakob Schwendner, Sylvain Joyeux, and Frank Kirchner. Using embodied data for localisation and mapping. *Journal of Field Robotics*, 2014. To appear.

Roland Siegwart and Illah Reza Nourbakhsh. *Introduction to Autonomous Mobile Robots*. MIT press, 2004.

Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

R.C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.

D. Spenneberg and F. Kirchner. Embodied Categorization of Spatial Environments on the basis of Proprioceptive Data. In *Proc. of the 3rd International Symposium on Adaptive Motion in Animals and Machines*, volume 3, page 03, 2005.

J. Sprickerhof, A. Nüchter, K. Lingemann, and J. Hertzberg. A heuristic loop closing technique for large-scale 6d slam. *Automatika–Journal for Control, Measurement, Electronics, Computing and Communications*, 52(3), 2011.

C. Stachniss, D. Hahnel, and W. Burgard. Exploration with active loop-closing for FastSLAM. In *Proc. IEEE IROS*, 2004.

M.J. Swain and D.H. Ballard. Indexing via color histograms. In *Proc. Third International Conference on Computer Vision*, pages 390–393, 1990.

S. Thrun. The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *The International Journal of Robotics Research*, 25(5-6): 403–429, May 2006. doi: 10.1177/0278364906065387.

Sebastian Thrun and A Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the National Conference on Artificial Intelligence*, 1996.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

N. Trawny, A.I. Mourikis, S.I. Roumeliotis, A.E. Johnson, and J.F. Montgomery. Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks. *Journal of Field Robotics*, 24(5):357–378, 2007.

R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. IEEE IROS*, 2006.

Christian Weiss, Nikolas Fechner, Matthias Stark, and Andreas Zell. Comparison of different approaches to vibration-based terrain classification. In *European Conference on Mobile Robots (ECMR)*, 2007.

Kai M. Wurm, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1217–1222. Ieee, October 2009.

K.M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, 2010.