Hochschule für Technik und Wirtschaft Dresden (FH)
Fakultät Informatik/Mathematik

Diplomarbeit

im Studiengang Informatik

Thema: Localization of autonomous ground vehicles in dense urban environments

eingereicht von:	Marian Himstedt
eingereicht am:	27.11.2010
Betreuer:	Dr. Alen Alempijevic
	Prof. Dr.-Ing. habil. Hans-Joachim Böhme

Eidesstattliche Erklärung

Ich, Marian Himstedt, versichere, dass ich die Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Marian Himstedt

Zusammenfassung

Die Lokalisierung autonomer Fahrzeuge in dicht besiedelten städtischen Umgebungen ist problematisch. Klassische Anwendungen aus den Bereichen der Robotik und Automobilindustrie setzen auf die Verfügbarkeit von GPS Systemen, um ihre Position zu bestimmen. Aufgrund dichter Bebauungen erlaubt der Einsatz von GPS-Systemen keine verlässliche Lokalisierung in städtischen Umgebungen. Aus diesem Grund müssen alternative Ansätze verfolgt werden. Im Rahmen dieser Arbeit werden probabilistische Ansätze untersucht, die mit Hilfe der Odometrie des Fahrzeugs sowie einer monokularen Kamera eine Lokalisierung ermöglichen. Im Speziellen wird ein Verfahren vorgestellt, das versucht, visuelle Merkmale der Umgebung wahrzunehmen. Dazu wird zunächst eine topologische Karte anhand von Referenzorten aus der Umgebung gebaut, wobei jedem Referenzort eine Menge von visuellen Merkmalen zugeordnet wird. Durch die Anwendung einer Merkmalsselektion wird sichergestellt, das ähnliche Referenzorte klarer von einander unterschieden werden können. Mithilfe von Satellitenbildern und Daten aus geografischen Informationssystemen (GIS) wird die topologische Karte zu einer hybriden Umgebungsrepräsentation erweitert. Die Lokalisierung wird im Sinne der Wiedererkennung bekannter Referenzorte durchgeführt. Ein Partikelfilter wird verwendet, um wahrgenommene visuelle Umgebungsmerkmale mit der Fahrzeugodometrie zu fusionieren. Das implementierte System wird durch verschiedene Experimente evaluiert. Diese werden in dicht besiedelten städtischen Umgebungen, die durch hohe Dynamik und komplexe Gebäudestrukturen geprägt sind, durchgeführt.

Abstract

The localization of autonomous ground vehicles in dense urban environments poses a challenge. Applications in classical outdoor robotics rely on the availability of GPS systems in order to estimate the position. However, the presence of complex building structures in dense urban environments hampers a reliable localization based on GPS. Alternative approaches have to be applied In order to tackle this problem. This thesis proposes an approach which combines observations of a single perspective camera and odometry in a probabilistic framework. In particular, the localization in the space of appearance is addressed. First, a topological map of reference places in the environment is built. Each reference place is associated with a set of visual features. A feature selection is carried out in order to obtain distinctive reference places. The topological map is extended to a hybrid representation by the use of metric information from Geographic Information Systems (GIS) and satellite images. The localization is solved in terms of the recognition of reference places. A particle filter implementation incorporating this and the vehicle's odometry is presented. The proposed system is evaluated based on multiple experiments in exemplary urban environments characterized by high building structures and a multitude of dynamic objects.

Contents

1 Introduction **1**

1.1 Statement of Problem 2

1.2 The Vehicle and Sensors 3

1.3 Structure of the Thesis 5

2 Background **6**

2.1 Ackerman Steering 6

2.2 Probabilistic State Estimation 7

2.2.1 Dynamical Systems 8

2.2.2 Bayesian Filter 9

2.2.3 Extended Kalman Filter 11

2.2.4 Particle Filter 13

2.3 RANSAC 15

2.4 Camera Geometry 17

2.5 Two-camera Geometry 18

2.5.1 Epipolar Geometry 18

2.5.2 Fundamental Matrix 18

2.5.3 Fitting a Fundamental Matrix using RANSAC 20

2.6 Visual Features 21

2.6.1 Scale Invariant Feature Transform 22

2.6.2 Matching of Feature Descriptors 26

3 Map Generation **27**

3.1 From Topological to Hybrid Map Representations 27

3.2 Feature Extraction and Selection 32

3.3 Practical Considerations and Limitations 36

4 Localization **38**

4.1 Motion Model 38

4.2 Observation Model 39
4.2.1 Simple Matching 40
4.2.2 Histogram based Approach 41
4.3 Resampling 44

5 Experimentation 46
5.1 Map building 46
5.2 Experiment 1 47
5.3 Experiment 2 48
5.4 Experiment 3 48
5.5 Summary 49

6 Discussion 53
6.1 Conclusion 53
6.2 Limitation 53
6.3 Future work 54

A Appendix 57
A.1 Discrete Convolution in 2D 57
A.2 Particle Filter Implementation 58
A.3 Theses 59

Bibliography 60

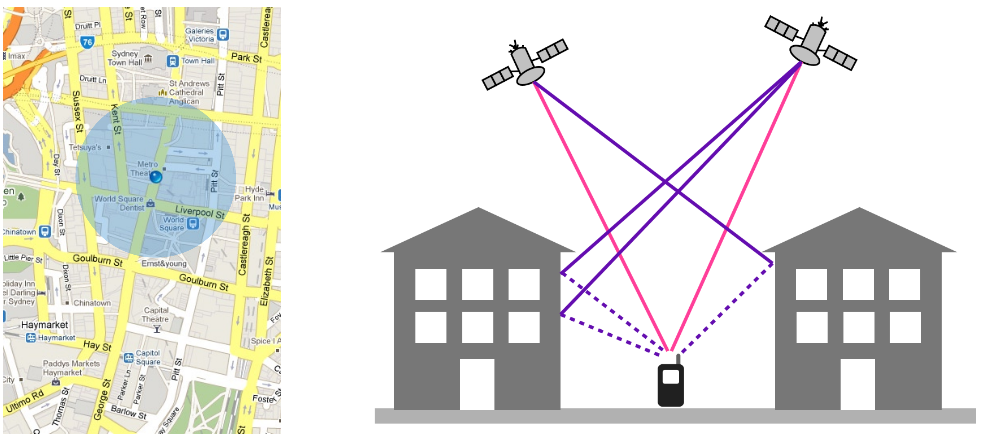
1 Introduction

The interest in autonomous ground vehicles has significantly increased in the past years. The DARPA, an agency for the United States Department of Defence, invited research teams from all over the world to participate in challenges for autonomous vehicles, namely the Grand Challenges and the Urban Challenge[1]. These attracted a lot of attention in several research communities, particularly in mobile robotics and automotive as many of their research interests are fused in the development of autonomous vehicles. Furthermore, the publicity has taken notice of this development. Latter, however, has a ambivalent view regarding this. One group appreciates and follows this field of research with interest, others rather have fears. Objectively one could argument that the better the autonomous operation of vehicles the more they can assist in crucial manoeuvres. In this way there have established a lot of assistance systems supporting the driver, for instance, in keeping the lane, adapting the speed in accordance with the traffic density or checking the blind spot when changing the lane. However, there is still a large need for research. One of the major problems in the DARPA Urban Challenge was the vehicle localization. Traditionally the localization in outdoor robotics as well as automotive applications relies on GPS in combination with a prior map. This generally enables satisfying results in rural areas. In contrast, the operation in dense urban environments poses a huge challenge for autonomous vehicles. The presence of complex building structures causes unreliable GPS position estimates. Thus, alternative approaches are necessary in order to estimate the vehicle's position. This thesis addresses exactly that problem. The localization is carried out using an onboard single camera, wheel encoder readings as well as steering angle measurements. Due to the use of monocular vision a model working in the space of appearance is selected. Hence a prior map including reference places of the environment is learnt. Moreover a probabilistic framework capable of incorporating different sensors is introduced. To be more specific, a particle filter is applied.

[1]More information can be found on: http://www.darpa.mil/grandchallenge/

1.1 Statement of Problem

The localization of vehicles in rural and interurban areas based on GPS works quite reliable. That is because the field of view with reference to the sky is not blocked. In those situations the use of GPS is suitable especially because of its ease and its good incorporation with existing digital maps. However, using GPS in dense urban environments is crucial due to the presence of so called urban canyons as illustrated in Figure 1.2. The accuracy of the position estimate based on GPS highly depends on the satellite constellation which is expressed as the dilution of precision (DOP)[31]. The precision in urban canyons is degraded as the view to the sky is significantly reduced. Hence the satellites that can be perceived from the GPS receiver are arranged very tight. The position estimate and its uncertainty in an urban canyon is exemplarily illustrated in Figure 1.1a. This uncertainty can even lead to a wrong position estimate on street level as an adjacent street might be selected. The DOP can be estimated based on the current satellite constellation. That means this uncertainty is known at each time. However, GPS measurements in urban canyons are also subject to other sources of error. Signals from satellites are not necessarily received on the direct path as buildings might reflect them. Thus, these signals travelled a further distance than expected by the receiver. This is known as multipath error and more difficult to estimate [31, chapter 5]. This often leads to a "jumping" of the estimated position. A typical scenario is illustrated by Figure 1.1b.



(a) *Position uncertainty due to satellite constellation*

(b) *Multipath error*

Figure 1.1: *Position estimation errors in urban canyons*

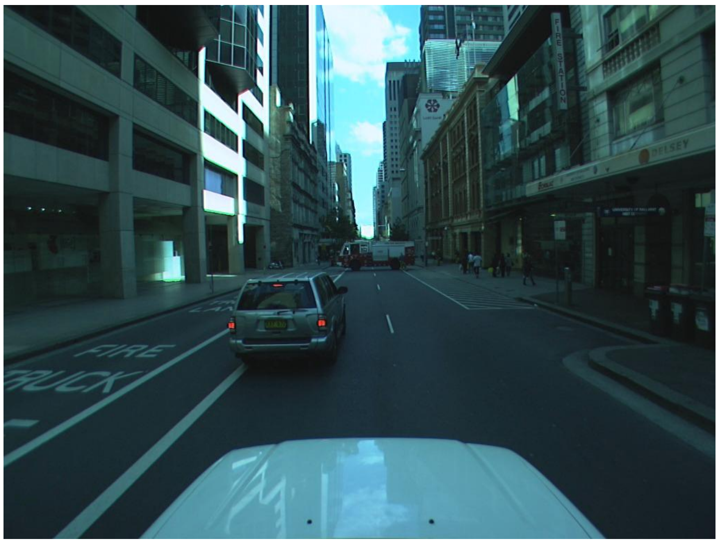


Figure 1.2: *Typical urban canyon scenario*

In order to achieve position estimates with higher precision other sensors have to be used. This thesis focuses on the localization using a single camera and odometry. Since there is no information about the depth to landmarks in the scene, we present a localization approach working in the space of appearance. Thus, we build a map containing visual features of different places in the environment. As the distance between places are kept small, ambiguities in the recognition of places have to be minimized. Hence a feature selection is carried out. The topological structure of these places is extended to a hybrid map representation by the use of local distance measurements and ground truth information obtained from Graphical Information Systems (GIS) and satellite images. Localization is carried out based on this map. Generally, the localization problem can be divided into two different classes. In the first class, it is assumed that the initial position of the vehicle is known. The goal of the localization is to track this position. In the other class, there is no information about the initial position. In literature, this is referred to as global localization [27]. Latter is addressed in this thesis.

1.2 The Vehicle and Sensors

The vehicle used for our experimentation is a Ford Courier with front wheel drive (see Figure 1.3).

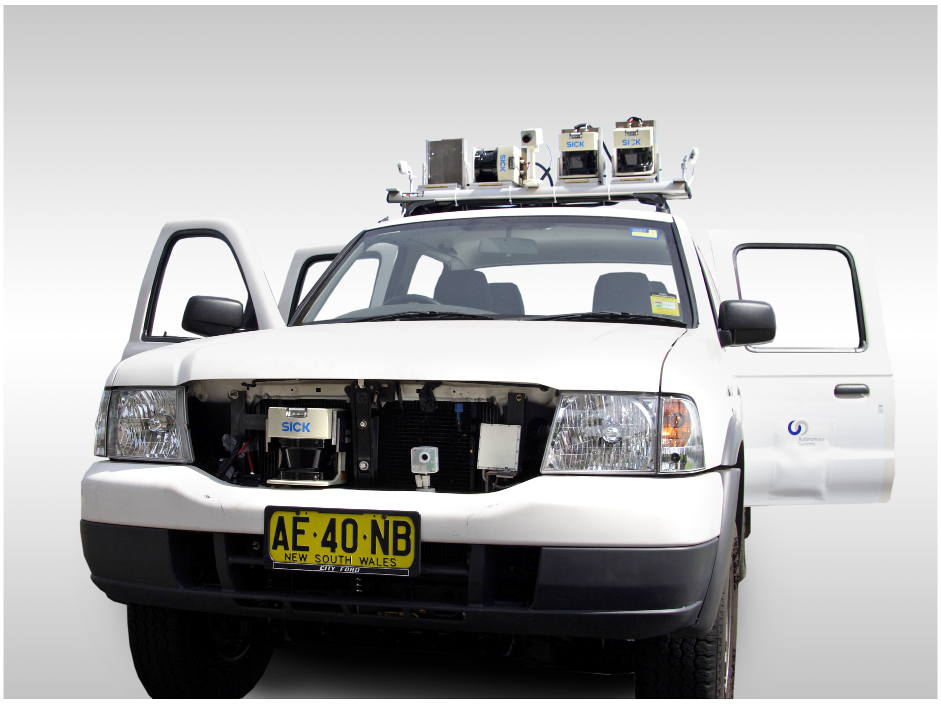


Figure 1.3: *Front view of the vehicle*

It posses the following sensors:

- 3 SICK LMS-291 laser range finder (two on the roof and one at the front)
- a UWB radar at the front
- Hitachi HV-F31 (monocular colour camera)
- 4 wheel encoder (part of the ABS system)
- steering angle measurement unit
- Crossbow DMU FOG IMU700CA inertial measurement unit (IMU)
- DGPS unit

Moreover the vehicle is equipped with two standard personal computers that are located in the trunk (see Figure 1.4). The camera used in our experiments is a Hitachi HV-F31 (see Figure 1.5). It provides images with a resolution of 1024×768 and a frame rate up to 7.5 fps.

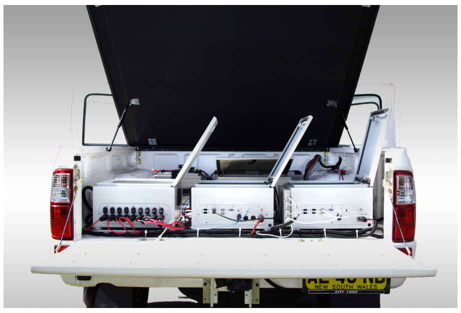


Figure 1.4: *Trunk of the vehicle*

Figure 1.5:
Experimentation camera. Hitachi
HV-F31

1.3 Structure of the Thesis

The remainder of this document is organised as follows.

Chapter 2 discusses the theoretical background.

Chapter 3 addresses the problem of map acquisition.

Chapter 4 presents the implemented localization approach.

Chapter 5 contains experimental results.

Chapter 6 concludes this thesis and motivates future work.

2 Background

2.1 Ackerman Steering

Each vehicle has a certain drive system incorporating its specific physical characteristics. Typical systems in mobile robotics are differential, tricycle and synchro drives. As mentioned at the beginning of this thesis our work focusses on cars. Thus, we want to introduce a drive system applied for automotive vehicles: the Ackerman steering. When a car follows a path around a curve, its front wheels have to rotate. If both front wheels rotate about identical angles the tires are subject to systematic sideways slip [22]. In order to avoid this effect the inner front wheel has to rotate about a slightly sharper angle than the outer wheel when turning. The geometrical correct solution for all wheels adjusts the axles as radii of a circle with a common centre, the Instantaneous Center of Rotation (ICR). Since the rear wheels are fixed, the ICR lies on a line extended from the rear axle (see Fig. 2.1). Applying this principle, a vehicle is able to move instantaneously along a circle. Thus, it satisfies the Ackerman equation [15]:

$$\cot\theta_i - \cot\theta_o = \frac{d}{l}, \tag{2.1}$$

θ_i, θ_o being the relative steering angles of the inner wheel and the outer wheel respectively, l the longitudinal and d the lateral wheel separation. For further applications we would like to determine the vehicle's steering angle ϕ being relative to the vehicle's heading [15]. Thus, an imaginary center wheel is located at the point R. The steering angle ϕ can be expressed using either θ_i or θ_o:

$$\cot\phi = \frac{d}{2l} + \cot\theta_i \tag{2.2}$$

$$\cot\phi = \cot\theta_o - \frac{d}{2l}. \tag{2.3}$$

In order to estimate a motion based on Ackerman steering, we also need the vehicle's velocity v_c at the center. As the rear axle is fixed, v_c can be calculated using the velocities v_{rl} and v_{rr} of the back wheel encoders [15]. The velocities v_{rl} and v_{rr} are proportional to their radii to the ICR [22]. Thus, the velocity v_c at the centre of rear axle can be estimated as:

$$v_c = (v_{rl} + v_{rr})/2 \tag{2.4}$$

Steering angle measurement units usually obtain the vehicle's heading direction as angular velocity ω_c instead of angular displacements as ϕ.

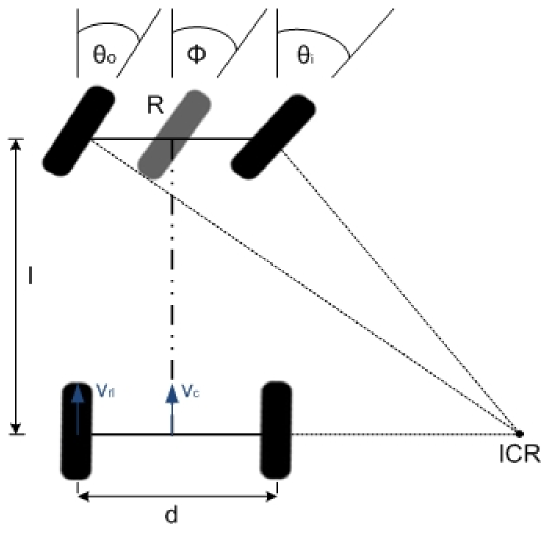


Figure 2.1: *Ackerman Steering*

2.2 Probabilistic State Estimation

In this section, we introduce a theoretical framework which allows inference in dynamical systems. In this way we model the dynamical system as a hidden Markov model. Firstly, we will give a brief introduction to dynamical systems. Afterwards the Bayesian filter which enables us to approximate unknown states in a probabilistic manner is presented. In addition to that we will deal with two implementations of the Bayesian Filter, namely the Kalman filter and the particle filter. The theory presented in this section is mainly based on Thrun et al. [27].

2.2.1 Dynamical Systems

A dynamical system can be considered at discrete time steps $t_1, ..., t_n$. For each time step t, the system executes an action u_t and receives an observation z_t. Both parameters are known respectively observable by the system. However, the system is unable to observe its actual state x_t. Thus, the estimation of the system state x_t is the goal in probabilistic state estimation. Actually, we approach to determine a belief $bel(x_t)$ about the state x_t. A dynamical system as described above is exemplarily shown in Figure 2.2.

In terms of vehicle localization the state x_t is the pose which contains the vehicle's position and orientation. An action u_t can be described as a motion of the vehicle measured by onboard odometers and steering angle sensors. Readings from vehicle's sensors, for instance laser range finder or cameras, are observations z_t. The belief $bel(x_t)$ depends on all actions $u_1, ..., u_t$ and all previous observations $z_1, ..., z_t$. Hence, $bel(x_t)$ can be expressed in terms of a conditional probability distribution

$$bel(x_t) := p(x_t | u_{1:t}, z_{1:t}). \tag{2.5}$$

The abbreviations $u_{1:t}$, $z_{1:t}$ denote $u_1, ..., u_t$ and $z_1, ..., z_t$ respectively. This notation will be used from now on. In order to estimate the belief distribution, we need some a priori knowledge. To be more precisely, we have expectations of what will happen if the vehicle performs action u_t. Ideally, this could be expressed as a function, namely a state transition function

$$g(x_t, u_t) = x'_t, \tag{2.6}$$

propagating the system to a new state x'_t given state x_t and action u_t. In real-world applications, actions often do not result in the desired state. For instance, our vehicle's wheels might be exposed to slipping on the road. In order to incorporate this kind of noise, we model our state transition as state transition probability,

$$p(x_{t'} | x_t, u_t), \tag{2.7}$$

which can be understood as the conditional probability distribution of our new state x'_t given state x_t and action u_t. With respect to vehicle localization, the state transition probability can be referred to as motion model. In addition to the action u_t we are given an observation z_t. What can we infer from z_t about the system state

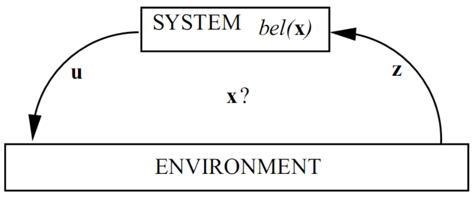


Figure 2.2: *Dynamical System*

x_t? Ideally, we would like to directly access the conditional probability distribution of x_t given the observation z_t.

$$p(x_t|z_t) \tag{2.8}$$

Obtaining this distribution directly is often impossible. This is why, the counterpart, the probability of z_t given x_t is used:

$$p(z_t|x_t) \tag{2.9}$$

This distribution poses the observation model of our state estimation. Providing our state space is discrete, the observation model can be estimated by setting x_t to different possible states. Afterwards we sum how often an observation is obtained in a particular state. Thus, we can estimate the distribution $p(x_t|z_t)$ based on the observation model using Bayes' rule,

$$p(x_t|z_t) = \frac{p(z_t|x_t)p(x_t)}{\sum p(z_t|x'_t)p(x'_t)}. \tag{2.10}$$

2.2.2 Bayesian Filter

This section deals with the Bayesian Filter which poses the most general method for state estimation. It enables the opportunity to recursively infer the belief $bel(x_t)$ given an observation model $p(z_t|x_t)$ and a state transition probability $p(x_t|u_t, x_{t-1})$. Firstly, the belief $bel(x_t)$ can be expressed as the probability distribution of the state x_t given all previous actions u_t and observations z_t (see Equation 2.5). Applying Bayes' rule we can divide the posterior belief $bel(x_t)$ into the likelihood $p(z_t|x_t, u_{1:t}, z_{1:t-1})$ and the prior $p(x_t|u_{1:t}, z_{1:t-1})$,

$$bel(x_t) = \eta \cdot p(z_t|x_t, u_{1:t}, z_{1:t-1}) \cdot (x_t|u_{1:t}, z_{1:t-1}) \tag{2.11}$$

with η being a normalization factor. We can assume that the observation z_t is conditionally independent of all previous observations $z_{1:t-1}$ and actions $u_{1:t}$ given the state x_t. This is known as Markov property [27] (see also Figure 2.3). Thus, the likelihood can be reduced as follows:

$$p(z_t|x_{1:t-1}, u_{1:t}, z_{1:t-1}) = p(z_t|x_t). \tag{2.12}$$

Applying the law of total probability we can expand the prior $p(x_t|u_{1:t}, z_{1:t})$:

$$p(x_t|u_{1:t}, z_{1:t}) = \int p(x_t|u_{1:t}, z_{1:t}, x_{t-1})p(x_{t-1}|u_{1:t}, z_{1:t})dx_{t-1} \tag{2.13}$$

and obtain the following expression for our belief:

$$bel(x_t) = \eta \cdot (z_t|x_t) \cdot \int p(x_t|u_{1:t}, z_{1:t-1}, x_{t-1})p(x_{t-1}|u_{1:t}, z_{1:t-1})dx_{t-1}. \tag{2.14}$$

As mentioned above, the Bayesian filter targets a recursive structure which becomes apparent as we include the conditional probability of x_{t-1}. However, since the Markov property can be applied again, the term will be reduced further. Given the current action u_t and the previous state x_{t-1}, the state x_t becomes conditionally independent of all previous actions $u_{1:t-1}$ and observations $z_{1:t-1}$. Hence our belief becomes:

$$bel(x_t) = \eta \cdot p(z_t|x_t) \cdot \int p(x_t|u_t, x_{t-1})p(x_{t-1}|u_{1:t}, z_{1:t-1})dx_{t-1}. \tag{2.15}$$

Moreover, the state x_{t-1} is independent of u_t (see 2.3). This is why, the term $p(x_{t-1}|u_{1:t}, z_{1:t-1})$ reduces to $p(x_{t-1}|u_{1:t-1}, z_{1:t-1})$. With regards to Equation (2.5) we can express $bel(x_{t-1})$ as follows:

$$bel(x_{t-1}) = p(x_{t-1}|u_{1:t-1}, z_{1:t-1}). \tag{2.16}$$

Hence, we have the final recursive update rule of the Bayesian filter:

$$bel(x_t) = \eta \cdot (z_t|x_t) \cdot \int p(x_t|u_t, x_{t-1})bel(x_{t-1})dx_{t-1}. \tag{2.17}$$

As it can be seen in Eq. 2.15 the estimate of $bel(x_t)$ requires the calculation of an indefinite integral. If we assume that x_t is a state in a discrete space, the second term can be expressed in terms of a sum instead of an integral. All discrete probabilities sum up to one which is why the normalization step can be done afterwards. Thus the unnormalized beliefs $bel'(x_t)$ for each x_t are determined as follows:

$$bel'(x_t) = p(z_t|x_t) \cdot \sum_{x_t} p(x_t|u_t, x_{t-1}) bel(x_{t-1}). \tag{2.18}$$

the normalization step results in our belief $bel(x_t)$:

$$bel(x_t) = \frac{bel'(x_t)}{\sum_{\bar{x_t}} bel'(\bar{x_t})} \tag{2.19}$$

Summarizing it can be said that the Bayesian filter enables a recursive estimate of the belief $bel(x_t)$ which uses only the previous belief $bel(x_{t-1})$. Beliefs of further preceding steps are not considered. The principle of the Bayesian filter is the basis for further implementations. Typical derivates are the Kalman filter and the particle filter which are presented in the following sections.

2.2.3 Extended Kalman Filter

The extended Kalman filter (EKF) is an algorithm which is able to cope with probabilistic state estimation as we discussed in preceding sections. The Kalman filter implements the Bayesian filter in continuous space. This is done by modelling probability distributions by a Gaussian density function. In contrast to the initial version of the Kalman filter [16], the extended Kalman filter can deal with non-linear state transition and observation models. The belief distribution over an N-dimensional state vector $x = (x_1, x_2, ..., x_N)^T$ is expressed as a multivariate Gaussian $N(\mu, \Sigma)$ with the probability density function f_N:

$$f_N(x, \mu, \Sigma) = det(2\pi\Sigma)^{-\frac{1}{2}} exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)), \tag{2.20}$$

with μ being an N-dimensional mean vector and Σ an N $\times$ N-dimensional covariance matrix. We assume that the state transition probability $p(x'_t|x_t, u_t)$ can be expressed as a differentiable transition function g:

$$x'_t = g(u_t, x_t) + \epsilon_g, \tag{2.21}$$

where ϵ_g denotes a Gaussian noise with zero mean. This transition model is only an approximation. When propagating from state x_t to x'_t given an action u_t, the dynamical system is exposed to uncertainty. This is the reason why the noise ϵ_g is added. This noise can, for instance, occur due to wheel slipping during a motion

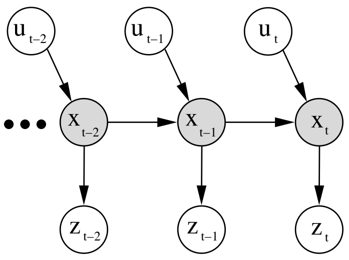


Figure 2.3: *Dynamical system as Hidden Markov Model*

of a vehicle. Moreover, the observation model $p(z_t|x_t)$ is defined as a differentiable function h:

$$z_t = h(x_t) + \epsilon_h, \tag{2.22}$$

Also, the observation model is driven by a noise ϵ_h . The cycle of the EKF is shown in Algorithm 1. At each cycle t the EKF is given an action u_t, an observation z_t and the previous belief $bel(x_{t-1}) = N(\mu_{t-1}, \Sigma_{t-1})$, where Σ_{t-1} models the system uncertainty. Within the state prediction step, we estimate the mean $\bar{\mu}_t$ of the prediction based on the state transition function g given the previous mean μ_{t-1} and the action u_t. The transition function g is not directly applied to the covariance of the prediction step $\bar{\Sigma}_t$. Instead g is approximated with a linear function at μ_{t-1}. Thus, the Jacobian G_t can be defined as follows:

$$G_t := \frac{\delta g(\cdot, u_t)}{\delta \mu_{t-1}}. \tag{2.23}$$

We multiply the Jacobian G_t on both sides of the covariance Σ_{t-1}. The matrix R_t which poses the uncertainty in the state transition ϵ_g is added. Within the correction step, the current observation z_t is used to correct the prediction. The variable V in line 4 characterizes the variance of the observation z_t. The covariance $\bar{\Sigma}_t$ estimated in the prediction step is applied to the observation model. Therefore, we generate an approximation of h, the Jacobian H_t:

$$H_t := \frac{\delta h}{\delta \bar{\mu}_{t-1}}. \tag{2.24}$$

The variable Q denotes the uncertainty of the observation reflected by ϵ_h. Given the observation variance V we compute the so-called Kalman gain K in line 5. It determines how much confidence we have in our observation and in how far it is incorporated to estimate our belief [27, p. 43]. The so-called innovation is computed by subtracting the observation z_t from the predicted measurement $h(\bar{\mu}_t)$ in line 6. Then, we obtain the updated mean μ_t by adding the predicted mean $\bar{\mu}_t$ to the innovation which is weighted by the Kalman gain. Finally, the covariance Σ_t is updated (line 7). The extended Kalman filter can only represent uni-modal distributions. Thus, it is not possible to express multiple hypotheses. Also, an EKF might be unsuitable if underlying processes are highly non-Gaussian. Nevertheless, the EKF poses a computationally inexpensive implementation for state estimation and is able to cope with measurement noise. In case of highly non-Gaussian processes or multi-modal probability distributions, other implementations of the Bayesian filter should be considered. One possible derivate is presented in the following section.

Algorithm 1 The Extended Kalman Filter (EKF)

```
1: function ExtendedKalmanFilter(($\mu_{t-1}, \Sigma_{t-1}$), $u_t$, $z_t$)
2:     $\bar{\mu}_t = g(u_t, \mu_{t-1})$                         ▷ state prediction
3:     $\bar{\Sigma}_t = G_t \cdot \Sigma_{t-1} \cdot G_t^T + R_t$
4:     $V = (H_t \cdot \bar{\Sigma}_t \cdot H_t^T + Q_t)$
5:     $K_t = \bar{\Sigma}_t \cdot H_t^T \cdot V^{-1}$                  ▷ Kalman gain
6:     $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$                ▷ update
7:     $\Sigma_t = (I - K_t \cdot H_t) \cdot \bar{\Sigma}_t$
8:     return ($\mu_t, \Sigma_t$)
9: end function
```

2.2.4 Particle Filter

We already discussed possible ways to solve the state estimation problem in a probabilistic manner. The crucial point is how to deal with the continuous probability distributions of our state space (see Equation 2.15). The EKF can model continuous probability distributions with the restriction to uni-modal Gaussian distributions, as discussed in section 2.2.3. This section introduces to an implementation of the Bayesian filter which is able to cope with any probability distribution, namely the particle filter. The particle filter uses a set of samples, the particles, to estimate probability distributions. A particle k poses a state $x_t^{[k]}$ and is assigned a weight $w_t^{[k]}$[1]. All particles are sampled from the state transition distribution. The weights are assigned according to the observation model. Algorithm 2 gives a summary of all necessary steps of the particle filter. The set of K particles representing the previous belief $bel(x_{t-1})$, the action u_t as well as the observation z_t are given as input. It can be seen that the algorithm consists of three sub cycles. Beginning

[1]This property is often referred to as importance factor [27].

with the update cycle, we draw a sample $\bar{x_t}^{[k]}$ from the state transition probability distribution $p(x_t|u_t, x_{t-1})$ (line 3). This step is the prediction of the particle filter. Its implementation depends on the state transition probability. Similar to the EKF, a state transition function:

$$x_t^{[k]} = g(u_t, x_{t-1}^{[k]}) + \epsilon_g \tag{2.25}$$

The noise ϵ_g can be individually set for each particle. The set of samples $\bar{x}_t^{[1]}, ..., \bar{x}_t^{[K]}$ reflect the prior probability distribution. The observation is applied in line 4. Each weight $\bar{w}_t^{[k]}$ is assigned corresponding to the probability of the observation z_t given the hypothesis $\bar{x}_t^{[k]}$. As mentioned in 2.2.2, we know that all discrete probabilities sum up to one. This normalization step is done in line 7. Afterwards, the weighted particle set represents the belief $bel(x_t)$:

$$bel(x_t) \approx \sum_{k=1}^{K} w^{[k]} x^{[k]}. \tag{2.26}$$

The final step, the resampling, is the particularity of the particle filter. The challenging point is the replacement of the weighted particle set by an unweighted set without modification of the posterior distribution $bel(x_t)$. Particles indices from the posterior distribution are drawn (line 10). The particle $x_t^{[i]}$ is drawn with the probability reflected by the weight $w_i^{[k]}$. A new sample with the state $\bar{x}_t^{[i]}$ is initialized. As a result, particles with high weights are likely to be represented in many samples. Lower weighted particles are represented by fewer samples in contrast. Consequently, the particles are concentrated in regions with higher density of the posterior distribution $bel(x_t)$. Here, the posterior $bel(x_t)$ can be multi-modal since multiple hypotheses can be tracked. Note, that the size of the particle set highly determines the accuracy of the state estimate. Ideally, the number of particles approaches infinity. In this case the particle filter operates as if it was working with continuous probability distributions [1]. Only in this case we are able to exactly estimate our belief $bel(x_t)$, otherwise it is an approximation as defined by equation 2.26. For practical considerations, the size of particle set is a trade-off between computational complexity and accuracy. Thus, it highly depends on the specific application.

Algorithm 2 The Particle Filter

```
 1: function ParticleFilter($\{x_{t-1}^{[1]}, ..., x_{t-1}^{[K]}\}, u_t, z_t$)
 2:     for k=1 to K do                                        ▷ Update
 3:         sample $\bar{x}_t^{[k]}$ from $p(x_t|u_t, x_{t-1}^{[k]})$
 4:         $\bar{w}_t^{[k]} := p(z_t|x_t^{[k]})$
 5:     end for
 6:     for k=1 to K do                                        ▷ Normalization
 7:         $w_t^{[k]} := \bar{w}_t{}^{[k]}/(\sum_{i=1}^{K} \bar{w}_t^{[i]})$
 8:     end for
 9:     for k=1 to K do                                        ▷ Resampling
10:         draw $i$ with probability $\propto w_t^{[i]}$
11:         $x_t^{[k]} := \bar{x}_t^{[i]}$
12:     end for
13:     return $\{x_t^{[1]}, ..., x_t^{[K]}\}$
14: end function
```

2.3 RANSAC

Let us assume, we want to fit a model to a given data set S. This data set is driven by noise and hence contains a certain number of outliers. Outliers in this case denote samples that cannot be fitted to our desirable model and might influence the estimate in a negative manner. Fischler et. al. [11] presented the Random Sample Consensus (RANSAC) algorithm which tackles this problem. It can robustly fit a model even in the presence of many outliers. RANSAC suits particularly in those cases [13]. To start with, we consider the problem of fitting a line to a number of points in 2D (see Figure 2.4). Points must not deviate more than t units from the fitted line to be considered as inliers. Thus, a classification in terms of outliers and inliers based on the fitted line is done as well. The parameter t denotes the threshold which can be adjusted according to the noise. According to the RANSAC algorithm 2 points P_i and P_j are randomly selected from the data set. A line is determined by these points. Now, those points that are located within the distance threshold t are counted. The support of the sample P_iP_j is measured based on the number of inliers for this line. This process is repeated a number of times for randomly selected points. The sample P_iP_j with the highest support is considered as the robust fit. As result we obtain a fitted line and a number of inliers that can be associated with this model. The remaining points are classified as outliers. More generally, the fitting of a model based on RANSAC can be summarized as follows[11]:

1. A sample of s data points is randomly selected from S to instantiate the model.

2. Estimate the set of data points S_i that are located within the distance threshold t. The set S_i represents the inliers of S.
3. If the number of inliers is greater than a threshold T, the model is estimated based on all points in S_i and the algorithm terminates.
4. If the number of inliers falls below T repeat from step 1.
5. After N trials the sample with the highest support is selected. Based on the inliers for this sample, estimate the model and terminate.

The number of trials N can be determined as follows[13]:

$$N = \log(1-p)/\log(1-(1-\epsilon)^s), \tag{2.27}$$

where ϵ denotes the proportion of outliers in the data set. The parameter p expresses the probability that at least one of the samples of s points does not contain outliers.

The threshold T can be set according to the expected number of inliers. Assuming n data points, T could be set: $T = (1-\epsilon)n$.

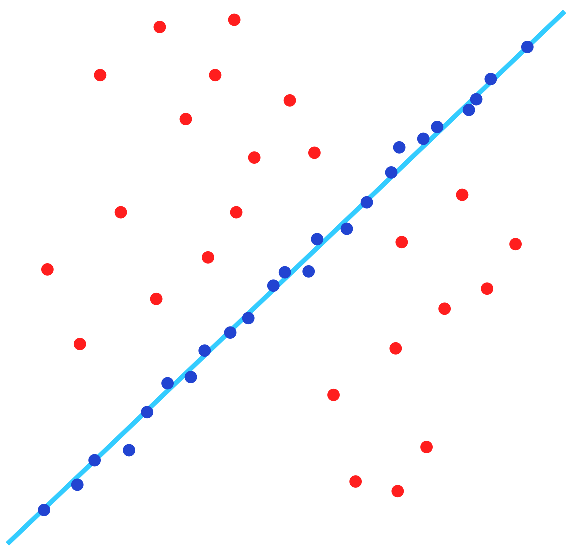


Figure 2.4: *A fitted line based on RANSAC*

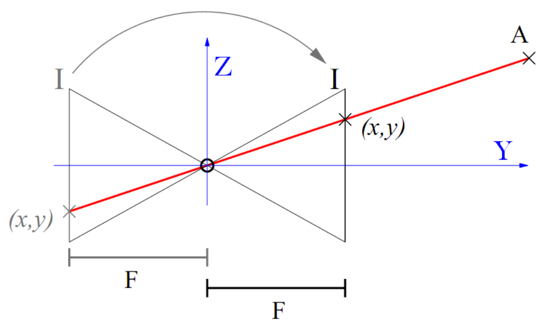
2.4 Camera Geometry

The simplest camera model is a pinhole camera. A pinhole camera can be imagined as a box that contains a small hole on one side and a screen respectively an image plain on the other side. Objects that are located in front of the camera are projected onto the screen upside-down. The image's scale depends on the distance of the hole to the projection screen which is called the focal length F. We assume that the image plain is placed in front of the camera. A point on the screen is called the pixel x and expressed as $x = (x, y)$. Thus, a pixel contains a horizontal and a vertical coordinate. When observing a point X with the camera-centric Cartesian coordinates $X = (X, Y, Z)$ in the scene, it is projected onto the image plain as follows (see also Figure 2.5a):

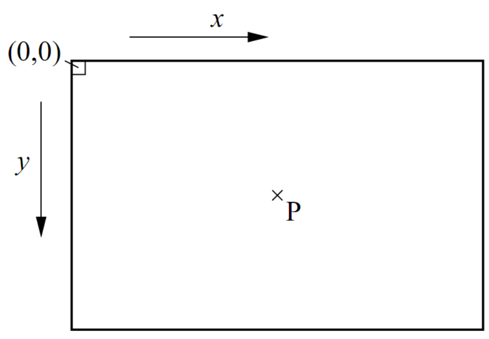
$$\hat{x} = \left(\frac{X \cdot F}{Y}; \frac{Z \cdot F}{Y}\right) \tag{2.28}$$

This actually assumes that the origin of the screen is its centre. According to conventions [13] , the origin $(0, 0)$ is placed at the top left pixel (see also Figure 2.5b). The centre of the screen is called the principal point $P = (P_x, P_y)$. Theoretically, P_x could express half the screen width and P_y half the screen height. However, in practice the principal point often deviates from that. Thus, the sign of y-coordinate is changed and the principal point is added:

$$x = \left(\frac{X \cdot F}{Y} + P_x; -\frac{Z \cdot F}{Y} + P_y\right) \tag{2.29}$$



(a) *Slice of the camera-centric (X,Y,Z) space. Image I is placed in front of the camera.*



(b) *Conventional denotation of image coordinates*

Figure 2.5: *Pinhole camera model*

2.5 Two-camera Geometry

2.5.1 Epipolar Geometry

After discussing the characteristics of a single camera, we consider two camera views in this section. The relationship of two camera views is specified by the epipolar geometry[2]. The application of two-view geometry does not necessarily require a stereo camera system. The case of two images captured by a single camera from different positions is also considered.

To start with, we place two cameras in a scene, where the first camera centre is at point O_1 and the second camera centre at O_2. The line b determined by O_1 and O_2 is called the baseline. The first camera observes the scene point X (see Figure 2.6) at the position x in its local camera coordinate system. The exact position of X is unknown. However, starting from the camera origin O_1 we can cast a ray r through x towards X. Thus, we know that X lies on the ray r. The epipolar plane π is defined by the baseline b and the ray r. It is not possible to predict that X will be visible at the position x' in the second frame given the first frame. Nevertheless, we know that the ray r' from O_2 through x' lies in π. Hence, x' is located on the line l' which poses the intersection of the second camera's image plane and the epipolar plane π. The line l' is a projection of the ray r into the second camera's frame. It is called the epipolar line. All epipolar lines intersect in a common point which is called the epipole e' [13, p. 241]. This calculation can also be done based on x' instead of x. Thus, we obtain the epipolar line l and the epipole e in the first camera frame. An algebraic representation of the epipolar geometry is discussed in the next section.

2.5.2 Fundamental Matrix

Given two images, we can say that for each point x in the first image, there is a corresponding epipolar line l' in the second image. If the point x' matches x, it has to be located on the line l'. Hence each valid pair of correspondences xx' has to satisfy the following condition:

$$x'^T F x = 0, \tag{2.30}$$

[2]The theory of this section mainly bases on Hartley et al. [13]

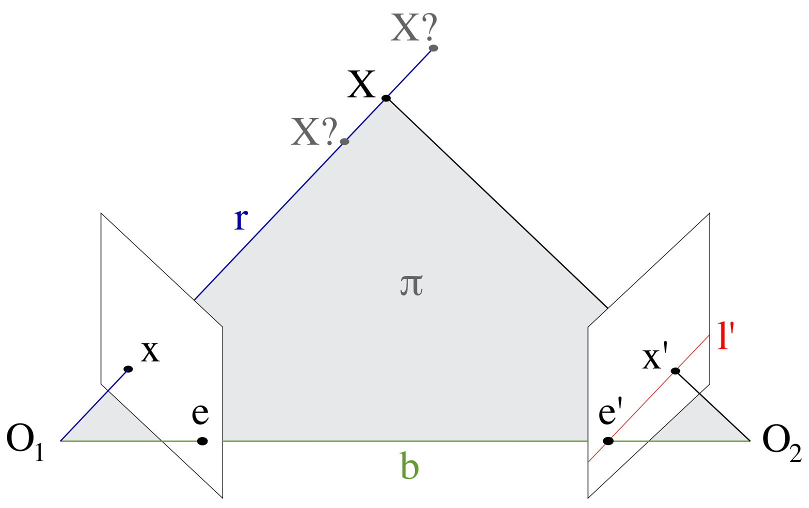


Figure 2.6: *Epipolar Geometry. The epipolar l' for the point X is shown in the second camera*

where F denotes the fundamental matrix. Thus, each map xl' is determined by F. The mathematical derivation for the fundamental matrix is given in Hartley et al. [13, p. 243 et seq.]. We give a brief overview of the estimation of the fundamental matrix based on the eight-point algorithm. First, we need a set of N correspondences xx' with $N \geq 8$. Points are set to $x = (x, y, 1)^T$ and $x' = (x', y', 1)^T$. The fundamental matrix does not require normalized image coordinates in terms of the camera's intrinsic parameters. However, as the image coordinates (x, y) can contain values in the range of 100 - 1000, they have to be normalized, such that $\hat{x} = Tx$ and $\hat{x}' = T'x'$, where T and T' denote normalizing transformations consisting of a translation and scaling. This normalization step significantly improves the accuracy of the estimation of the fundamental matrix [13].

Next, for each pair $\hat{x}\hat{x}'$, the following can be denoted:

$$\hat{x}'\hat{x}f_{11} + \hat{x}'\hat{y}f_{12} + \hat{x}'f_{13} + \hat{y}'\hat{x}f_{21} + \hat{y}'\hat{y}f_{22} + \hat{y}'f_{23} + \hat{x}f_{31} + \hat{y}f_{32} + f_{33} = 0. \tag{2.31}$$

Thus, we can summarize:

$$A = (\hat{x}'\hat{x}, \hat{x}'\hat{y}, \hat{x}', \hat{y}'\hat{x}, \hat{y}'\hat{y}, \hat{y}', \hat{x}, \hat{y}, 1) \tag{2.32}$$

$$A \cdot f = 0 \tag{2.33}$$

Applying this to N correspondences, a set of linear equations is obtained:

$$A \cdot f = \begin{bmatrix} \hat{x}'_1\hat{x}_1 & \hat{x}'_1\hat{y}_1 & \hat{x}'_1 & \hat{y}'_1\hat{x}_1 & \hat{y}'_1\hat{y}_1 & \hat{y}'_1 & \hat{x}_1 & \hat{y}_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hat{x}'_N\hat{x}_N & \hat{x}'_N\hat{y}_N & \hat{x}'_N & \hat{y}'_N\hat{x}_N & \hat{y}'_N\hat{y}_N & \hat{y}'_N & \hat{x}_N & \hat{y}_N & 1 \end{bmatrix} \cdot f = 0. \tag{2.34}$$

Using this set of homogenous equations, f can be determined up to scale using Singular Value Decomposition (SVD) with $SVDA = UDV^T$. The fundamental matrix $\hat{F}$ is determined by the smallest singular value of A and is of rank 2. The solution as described above does not necessarily satisfy this requirement. In order to correct this, the matrix $\hat{F}$ is replaced by the matrix $\hat{F}'$ which minimizes the Frobenius norm $\|\hat{F} - \hat{F}'\|$ with the condition $det\hat{F}' = 0$. Again, SVD is used to solve this. In the final step, F' is denormalized to ensure that it corresponds to the input points xx'. Thus, the the fundamental matrix F is obtained as follows:

$$F = T'^T \hat{F}' T \tag{2.35}$$

A more detailed explanation for the estimation of a fundamental matrix based on the eight-point algorithm can be found in Hartley et al. [13, chapter 11]. The eight point algorithm is the most basic implementation for the estimation of a fundamental matrix. Other algorithms require fewer correspondences while taking higher computational requirements into account. Nister [21] proposed an algorithm that enables the estimation based on five correspondences. When assuming planar motion the fundamental matrix can be calculated using only two correspondences [4]. Scaramuzza et al. [23] involved constraints for nonholonomic vehicles as cars to reduce the number of correspondences to one.

2.5.3 Fitting a Fundamental Matrix using RANSAC

In the previous section, the estimation of a fundamental matrix based on the eight-point algorithm was discussed. Based on that, we present an approach that fits a fundamental matrix given a number of correspondence points coming from two images. In this way, the RANSAC algorithm as discussed in Section 2.3 is applied. This approach is mainly inspired by Hartley et al. [13, p. 290 et seq.]. Restating the RANSAC algorithm, a distance measurement is required to distinguish inliers and outliers based on the sampled model. In terms of the estimation of a fundamental matrix, we have to measure how well two correspondences satisfy the epipolar constraint $x'^T Fx = 0$. Thus, we define a cost function based on a Sampson approximation:

$$\sum_i \frac{(x'^T_i F x_i)^2}{(Fx_i)_1^2 + (Fx_i)_2^2 + (F^T x'_i)_1^2 + (F^T x'_i)_2^2}, \tag{2.36}$$

where $(Fx_i)_j^2$ denotes the square of the j-th entry of Fx_i. Equation 2.36 is a first-order approximation to the geometric error. The derivation of this cost function is given by Hartley et al. [13, p. 287 et. seq].

The RANSAC based fitting of a fundamental matrix can be summarized as follows:

1. Interest points are computed in each image.
2. Correspondences are searched based on the similiarity of interest points.
3. RANSAC robust fitting
    a) A random sample of 8 correspondences is selected. A fundamental matrix F is estimated based on these points using the eight-point algorithm (see 2.5.2).
    b) The distances d of putative matches based on Equation 2.36 are calculated.
    c) The inliers consistent with the current fundamental matrix are estimated based on the number of correspondences satisfying the condition: $d < t$ pixels.
    d) If the number of inliers is greater than the threshold T, the currently estimated F is selected and the fitting procedure terminates. Otherwise we continue with step 3a

According to Hartley et al. [13] a final non-linear re-estimation for F based on all correspondences classified as inliers should be carried out. The Levenberg-Marquardt algorithm is recommended for this [13, section 11.6]. Note, that this step is only necessary if the fundamental matrix is used for further processing. As mentioned in section 2.3, a further reason for using RANSAC is the classification of outliers based on a fitted model.

2.6 Visual Features

An Image captured from a camera can be described as a high-dimensional matrix. In this section we will focus on how one can extract relevant information from images. To be more specific, we will deal with the extraction of relevant points of an image. We will refer to these points as visual features which usually consist of two components: an interest point and a descriptor. Feature detectors extract interest points with special characteristics as, for instance, high contrast. The characteristics of interest points are determined by their specific type of detector. In classical computer vision we distinguish between edge, corner respectively blob detectors. Once a keypoint is detected, the feature detector creates a descriptor characterizing its surrounding area. The descriptor vectors are compared to recognize features across multiple images. In addition to classical feature detectors there have established algorithms extracting highly distinctive features. Highly distinctive in this case denotes that given a set of features of an object or a scene we want to find corresponding features across a high dimensional database. For example, we capture images of different locations L of a city. For each location $l \in L$ we save a set of features. Having completed this training process a new image I around k ($k \in L$) is captured and extracted features from I are compared to features of all locations of L. As a result we should be able to determine that I was taken around k. Current state-of-the-art feature detectors enabling this are: Scale Invariant Invariant Feature Transform (SIFT) [18] and Speeded Up Robust Features (SURF) [2]. Those feature detectors have offered novel opportunities, for example, in object recognition, image retrieval systems and mobile robot localization. As SIFT is applied in the remainder of this thesis it is explained in detail in the following section.

2.6.1 Scale Invariant Feature Transform

The Scale Invariant Feature Transform (SIFT) detects features having the following characteristics:

- invariance to scale and rotation
- partially invariance to 3D viewpoint changes and illumination
- robust to affine distortion

The algorithm can be summarized in the following major steps:

1. **Scale-space extrema detection:** A scale space using a Difference of Gaussian (DoG) is searched to detect potential keypoints.
2. **Keypoint localization:** Location and scale of each candidate point are assigned. Keypoints are selected based on their stability.

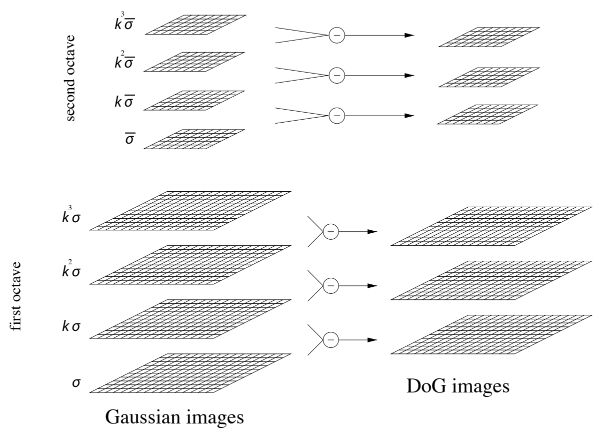


Figure 2.7: *Difference of Gaussian are computed from a pyramid of Gaussians*

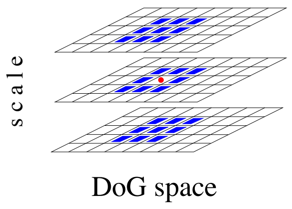


Figure 2.8: *A pixel (red) has 26 neighbors in DoG scale-space*

3. **Orientation assignment:** Each keypoint is assigned an orientation based on its local gradient. In case a keypoint has multiple local gradients, multiple instances of this keypoint are created with each having one orientation.
4. **Keypoint descriptor generation:** A descriptor is created based on local gradient information for each keypoint.

Interest points respectively keypoints detected by SIFT are local extrema in a scale-space [18]. Given an input image $I(x, y)$ and a scale σ the scale space can be defined as a function, $L(x, y, \sigma)$:

$$L(x, y, \sigma) = G(x, y, \sigma) \otimes I(x, y), \tag{2.37}$$

$\otimes$ being the convolution operation (see Appendix A.1) in x and y, and $G(x, y, \sigma)$ the Gaussian function:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{2.38}$$

By successively convolving I using $G(x, y, \sigma)$ of different scales σ a pyramid of Gaussian-blurred images is built (see Figure 2.7). This pyramid is divided into octaves. Each octave contains a fixed number of Gaussian images which differ by a constant scale factor k. Within each octave, adjacent Gaussian images are subtracted in order to get Difference of Gaussian (DoG) images. The scale space of a DoG image with scale σ and input image $I(x, y)$ can be defined as a function,

$D(x, y, \sigma)$:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) \otimes I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned} \tag{2.39}$$

Having completed octave O_i, the Gaussian Image of O_i with a variance of 2σ is subsampled. In this way every second pixel in each row and column is kept for the initial Gaussian image of octave O_{i+1}. Summarizing we can say that reducing the image size by an arbitrary scale factor is problematic. Thus, the image is scaled smoothly by convolving it with a Gaussian. In the next step, the 3-dimensional space of DoG images is searched for local extrema. A pixel only qualifies as local extremum if all adjacent pixel values are greater (local minimum) respectively smaller (local maximum). In the space of DoG images, each pixel has 26 neighbors, 8 in the current scale and 9 for the scale below as well as the scale above (see Figure 2.8). As a result a set of potential keypoints in the image is obtained. In the next step we determine a sub-pixel and sub-scale position (x, y, σ) by the interpolated location of the extrema. Keypoints having extrema with low magnitudes, and hence a low contrast, are rejected. Now, we determine the dominant orientation θ_{dom} of each keypoint. Using the keypoint's scale the Gaussian image, L, with the closest scale σ is selected. Then, gradients in a specific region around the keypoint are computed. For each candidate L(x,y,σ) of the selected scale σ we determine the magnitude, $|\nabla I(x, y)|$, and orientation, $\theta(x, y)$, of the gradient:

$$I_x(x, y) = L(x + 1, y) - L(x - 1, y) \tag{2.40}$$

$$I_y(x, y) = L(x, y + 1) - L(x, y - 1) \tag{2.41}$$

$$|\nabla I(x, y)| = \sqrt{I_x(x, y)^2 + I_y(x, y)^2} \tag{2.42}$$

$$\theta(x, y) = \arctan\left(\frac{I_y(x, y)}{I_x(x, y)}\right) \tag{2.43}$$

An orientation histogram is built from gradient orientations and magnitudes of an area around the keypoint location. The gradient magnitudes are weighted by a Gaussian centered around the keypoint location (see Figure 2.9). The significant peak in the histogram poses the dominant orientation θ_{dom} of the keypoint. If more than one major peak can be found, multiple instances of the keypoint having different dominant orientations θ_{dom} are created. Finally, each keypoint is assigned a descriptor. A SIFT descriptor is built based on local orientation histograms.

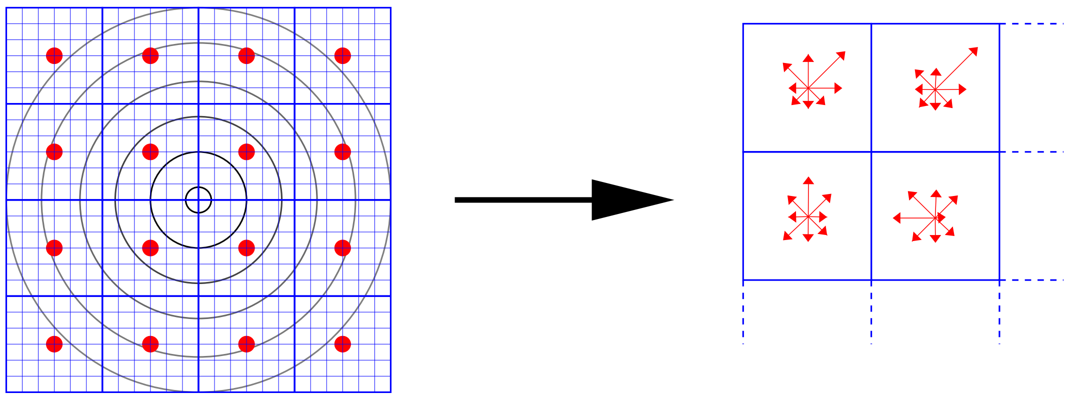


Figure 2.9: *SIFT descriptor. A gradient is computed for each pixel. The magnitudes of the gradients are weighted by a Gaussian. Afterwards, 16 orientation histograms are built out of 16 subregions.*

At the keypoint center we place a window which is divided into subregions. The magnitudes of the gradients are weighted by a Gaussian around the keypoint. Thus, gradient magnitudes being further away from the center are considered less to avoid sudden changes of the descriptor. The further away the gradient from the center the more likely it might be subject to misregistration errors [18]. An orientation histogram for each subregionis is built and placed in its center. In order to avoid boundary effects each gradient votes for an orientation in its adjacent histograms. The vote is weighted by $1 - d$, with d being the distance to the histogram. One of SIFT's characteristics is its rotation invariance. This is achieved by adapting the orientation histograms to the keypoint's dominant orientation θ_{dom}. Typical SIFT implementations use a window of $4 \times 4 = 16$ subregions, with each having an 8-bin orientation histogram. Hence, we get a 128-dimensional descriptor vector. One of SIFT's advantages is its partial invariance to change in illumination. This is obtained by the following steps. Fistly, the descriptor vector is normalized to unit length. In case of global changes in image contrast all pixels are multiplied by a constant factor. This effect can be avoided using vector normalization. If an image is subject to a global change in brightness, a constant is added to each pixel. As SIFT consideres local gradients, it is not affected by this. Both scenarios discussed assume linear changes in illumination. SIFT also considers non-linear changes, though incorporating those is more complex. These effects often occur due to constellations of shades irregularly influencing local gradients. Thus, magnitudes of gradients are changed by different amounts, orientations are typically influenced much less. SIFT attempts to minimize these effects by thresholding the values of the feature vector. In this way the influence of large gradients magnitudes is reduced while simultaneously increasing the influence of the orientations' distribution.

2.6.2 Matching of Feature Descriptors

We are given two sets of SIFT features, a and b, that are extracted from different images. In order to find correspondences, the descriptors of those features have to be compared. This is done by computing the nearest neighbours based on the Euclidian distances of the 128-dimensional descriptors. For each descriptor a_i of the first set, we calculate the distance $d(a_i, b_j)$ to each descriptors b_j of the second feature set. Each feature has only one correspondence in the other set. Whether two features match, is determined by the distance ratio of the nearest neighbour b_j and the second nearest neighbour b_k. Thus, the following condition has to be satisfied:

$$d(a_i, b_j) < \tau \cdot d(a_i, b_k), \tag{2.44}$$

where τ denotes the threshold. Adjusting this threshold often poses a problem as lower values might reject positive matches whereas higher values might accept more false positive correspondences. This threshold can be set more optimistically if the descriptor comparison is followed by geometric consistency checks based on the correspondences' locations in the image coordinate system. As result of the matching procedure we obtain a set of N corresponding features based on their similarity. For the remainder of the thesis we refer to the matching described above as the function *match*:

$$N = match(a, b) \tag{2.45}$$

Note that this is the basic procedure for matching descriptors. For larger databases of descriptors it is recommended to use, for instance, kd-trees. Efficient algorithms as Best Bin First could be applied to this. An extensive study about this is given by Schindler et al. [24].

3 Map Generation

This chapter deals with the generation of maps. In particular, the environment representations, needed for the localization methods in the following chapter, are presented.

As mentioned at the beginning of the thesis, our work focuses on localization and mapping using vision sensors, more precisely monocular vision. Our vehicle is also equipped with two dimensional laser range finders. However, employing those in dense urban environments in order to build a map is crucial as parking vehicles and other dynamic obstacles significantly block rays to building structures. Fortunately, this problem does not apply to a camera. The mapping in very dynamic urban environments poses a challenge though. Especially, the differentiation of dynamic and static objects in the environment is a difficult task. Often, one has to take into account that the map contains, for instance, parking vehicles. Alternatively, those objects could potentially be removed by a time-consuming post-processing step.

3.1 From Topological to Hybrid Map Representations

Using images of specific places, we build a map m of an urban environment. We can start off with a simplified case. While driving through an urban environment, images are continuously taken. Afterwards images of specific places could be selected. As the sequence of those images is known, a topology of reference places can be built. The topology can be closed if we see a reference place m_i, that has already been visited before. Providing this loop closure was carried out and also detected, we obtain a topological map[1]. An example of this is shown in Figure 3.1a. Each node of the topological map poses one selected reference place m_i of our trajectory. The

[1]An efficient algorithm for detecting loop closures in the space of appearance was proposed by Cummins et al. [9]

edge $e(m_i, m_{i+1})$ represents the two adjacent places m_i and m_{i+1}. Next, we could recognize our place with regards to our topological map, whenever we return to one of the reference places. One important advantage of topological maps is their compact representation. It can directly conduce to high level tasks as path planning. In addition to that it is close to the way how humans map environments [12]. However, using pure topological environment representations also implies distinct drawbacks. It assumes that places are clear distinctive. When travelling through very similar places, the missing metric information makes it hard to distinguish between these places. This could lead to unreliable position estimates with regards to the topology [20]. This is one of the main reasons for incorporating metric information.

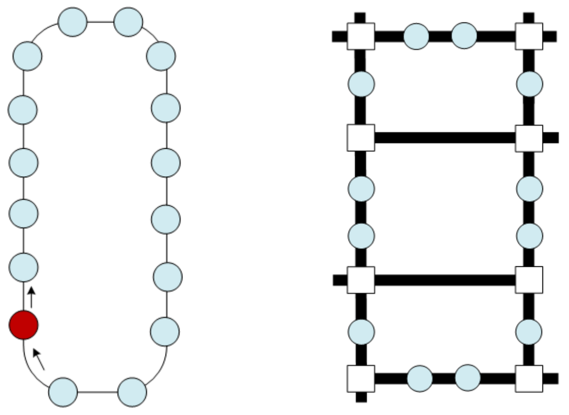

(a) *Simple topology closed at red node.* (b) *Map is extended by street courses.*

Figure 3.1: *Topological maps*

So far, we only considered the fact that images for reference places were continuously taken throughout our trajectory. Based on that, adjacency relations of reference places are expressed in form of edges. Now, this can be extended using a map of the urban environment which obtains further sparse information in form of streets. By comparing the trajectory to this street map, we can assign reference places an origin reference street. Streets can be divided into street segments whose ends are defined by two intersections. Thus, each reference place m_i is assigned a corresponding street segment. A map like this is illustrated by Figure 3.1b.

We still have the problem of differing two similar places. In addition to that, it is impossible to get position estimates, when remaining between two places. After leaving one reference place, we are uncertain about our whereabouts until we recognize the next reference place. In order to bridge this gap, metric information is included in our map. First, satellite images[2] of our trajectory are used to manually extract street courses as well as intersection markings (see Figure 3.2).

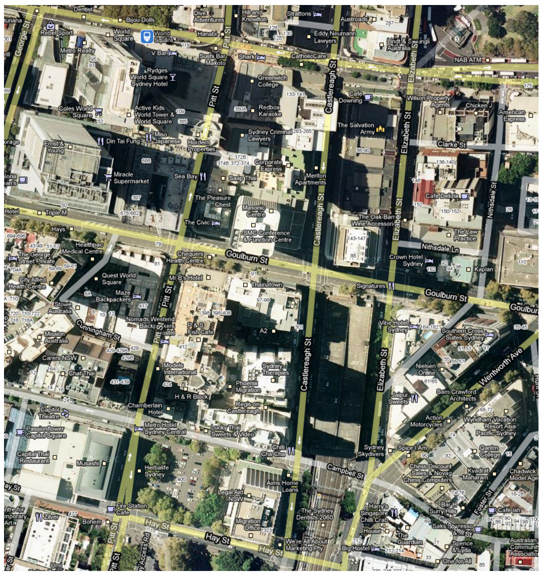
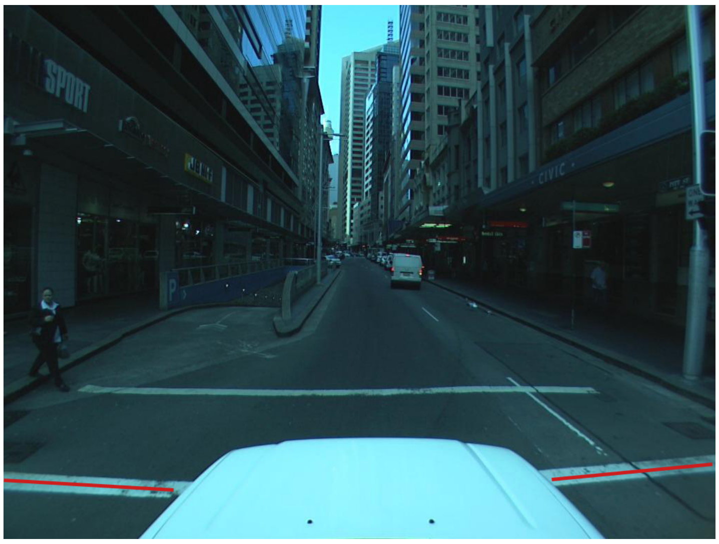


Figure 3.2: *Satellite image of an urban environment*

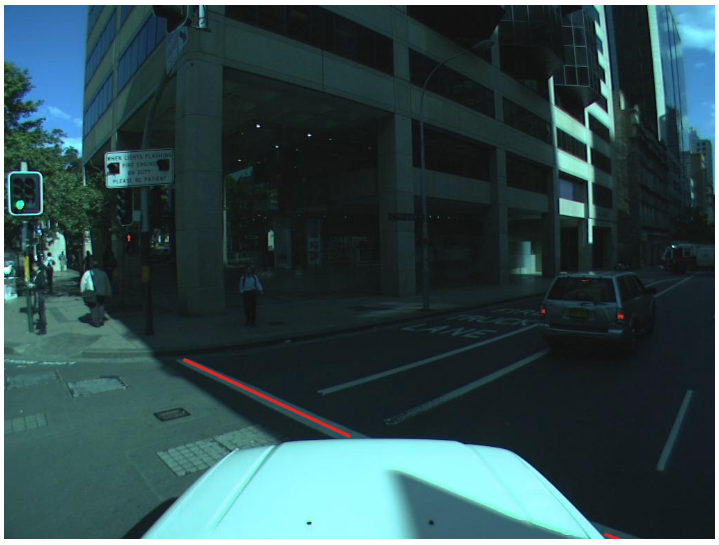
There are inner and outer intersection markings on Australian streets. We used the outer intersection markings as boundaries for street segments. Roads in dense urban environments typically have a straight course, whereas curved roads are very rare. This is also the case in our experimental course in the Central Business District of Sydney, Australia. Thus, we assume that each course of a street segment follows a fixed heading direction. This assumption is used for the remainder of the mapping process. The lengths of street segments that are manually extracted from satellite images often do not exactly correspond exactly to the ground truth. As each street segment is bordered by two intersection lines, they can be scaled if the distance between both lines is available. A geographical information system (GIS) is used in order to get these distance measurements which are provided by the New South Wales Road Transportation Agency. They can be considered as very accurate as they are obtained by total stations. We only incorporate the distances of two intersections lines, the width of the streets are not considered. Thus, streets have a fixed width.

In the next step, reference places of our trajectory are associated with this map. Hence, the entire image sequence is subdivided according to the street segments, where the first image of each segment is directly at the beginning intersection line (see also Figure 3.3a).

[2]Source of the satellite images is GoogleMaps: http://maps.google.com.au



(a) *Approaching intersection line while moving straight.*



(b) *Approaching intersection line while turning*

Figure 3.3: *Vehicle approaching intersection lines*

Starting from this image I_m, the distance to the following image I_{m+n} is estimated based on odometry. The distance $d(I_m, I_{m+n})$ travelled between I_m and I_{m+1} is calculated as follows:

$$d(I_m, I_{m+n}) = \int_{t_{I_m}}^{t_{I_{m+n}}} v_c(t)\, dt, \tag{3.1}$$

where t_{I_m} and $t_{I_{m+n}}$ are time steps of I_m and I_{m+n} respectively. As the vehicle's velocity v_c is sampled at discrete time steps t, we interpolate between wheel encoder time stamps and camera time stamps using cubic splines[3]. In this way distances between two reference places are estimated. Reference places correspond to a specific image. Based on the distance measurements and heading direction the reference places are assigned a location on the map with reference to the beginning intersection line. Note that the vehicle's heading direction is not taken into account. This is due to the fact that we assume that street segments have fixed heading directions. Using the odometry for distance estimates is crucial as errors in distance measurements accumulate. Thus, the error of the pose estimate based on odometry indefinitely increases over time [8]. This uncertainty must not be underestimated, even though only the translational component of the vehicle's motion is used. It increases with distance travelled along one street segment. However, errors are not accumulated through the entire map. As we begin from each street segment's intersection line which references directly to the map.

The reference place's location is also subject to another source of error. That is the selection of images that are closest to the intersection line. As the camera's frame rate is limited it is not always possible to select the frame which is exactly at the intersection line. This is even more crucial when the vehicle is turning into a street segment. This problem is illustrated in Figure 3.3b. Even though the distance measurements using odometry and the selection of corresponding image frames are subject to error, they obtain metric information for our map.

Based on the method described above, reference places for each street segment are selected every τ metres, where τ defines a minimum distance. This value should be chosen appropriately to avoid too high similarities of reference places. In our experiments this distance was typically set to $\tau = 10m$ or $\tau = 20m$.

Finally, non-street components of the satellite image are removed. Thus, we receive a map as it is illustrated by Figure 3.4.

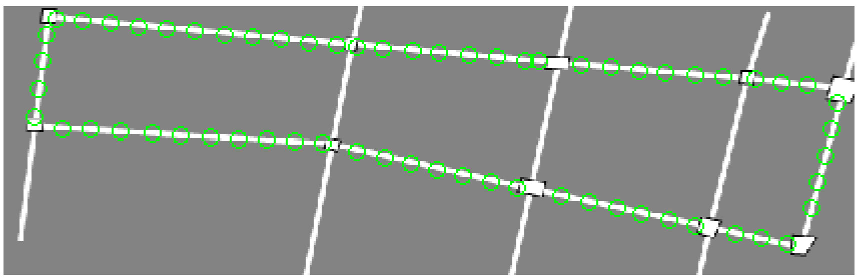


Figure 3.4: *Map based on GIS containing streets (white), intersection lines (black).*

[3]For more information about cubic splines, one might refer to [5, p. 43 et seq.].

3.2 Feature Extraction and Selection

In this section, we describe, how features are extracted and selected from images for reference places. Thus, we assume at this point that we are given a map m which is built as described in the previous section and contains the M reference places $m_1, ...m_M$. In addition to that we are given the images I that describe specific reference places.

Why do we need to select features? This is an important question. Many approaches [9, 24] that tackle the problem of localization in the space of appearance do not rely on feature selection. Actually, we could simply save all features of an image that represents a reference place. As we discussed in the previous section, dense urban environments are often characterized by long, straight street segments. In addition to that, there are a lot of high building structures present. This is the reason why one can observe similar structures from different places along a street even though they are far apart from each other. Hence, when extracting features from those places, there will be many features in common due to high building structures in the background. This makes it very hard to distinguish between places. Thus, we want to keep only those local features that are most descriptive for a place. Moreover, we try to remove features describing dynamic objects. Apart from the advantages mentioned, it is also important to store only features that are necessary to recognize a place. Thus, the requirements in terms of memory are reduced by a feature selection.

Our feature extraction and selection procedure can be divided into the following steps:

- extraction of local features
- outlier detection using RANSAC
- selection of close by features

The steps are explained in detail in the following.

Feature Extraction

At first, the image I_a, which describes the reference place m_i, is searched for visual features. Therefore, we apply Scale Invariant Feature Transform (SIFT) [18] as feature detector. As mentioned in section 2.6.1, each SIFT feature is described by the quadruple $\langle (x, y), \sigma, \theta_{dom}, f \rangle$, where (x, y) is the sub-pixel location, σ the scale, θ_{dom} the orientation and f the descriptor vector. For further processing steps, we also extract SIFT features from the previous image I_{a-1}.

The use of local features for the mapping in urban environments is advantageous. Global techniques like colour similarities as proposed by [12] are highly disturbed by the dynamics in urban street scenes. Local features in contrast can be detected even though the scene structure changed due to the presence of dynamic objects. By considering local gradients, descriptors are still quite distinctive even though scenes are very similar.

As features are extracted from the entire images, we have to reject those that are associated with the vehicle itself. Therefore, a binary mask is applied which defines the vehicle related region and the remaining part. The coordinates of all SIFT keypoints are compared to this mask. Those features that are within the vehicle's region are rejected. The mask is set once the camera is fixed to the vehicle's roof. In order to incorporate vibrations during the operation the vehicle's region of the mask is set slightly greater than actually necessary.

Outlier detection using RANSAC

In the next step, we look for correspondences in the feature sets extracted from I_a and I_{a-1}. According to the procedure explained in Section 2.6.2, the feature descriptors of I_a and I_{a-1} are matched. The number of matches N_a is obtained as result. By simply matching features across two images, the first feature selection is applied. That is, features that are detected more often are potentially more stable. Next, the feature correspondences from I_a and I_{a-1} are checked for geometric consistency. Using RANSAC we fit a fundamental matrix F to the given correspondences, as described in detail in section 2.5.3. The number of matches N_a reduces to $\tilde{N}_a$ with $\tilde{N}_a \leq N_a$. Thus, outlier in terms of false matching SIFT descriptors are rejected. In addition to that, feature correspondences originated from close by dynamic objects can be detected and rejected. Other vehicles or pedestrians crossing our vehicle's path are robustly detected as outliers, providing there are enough inliers, such as features around static objects, present. The detection of dynamic obstacles using RANSAC works well if those are close by. However, the feature correspondences of further away objects might lie close to the epipolar lines and hence threatened as inliers. Non moving objects, such as parking vehicles, cannot be distinguished from other features such as those around buildings in this way. However, there is a possibility to learn places over longer periods. Given an image of the same place at a completely different time, one could estimate feature correspondences based on those. This would significantly improve the place recognition since it is unlikelier that a dynamic object appears at the same place again.

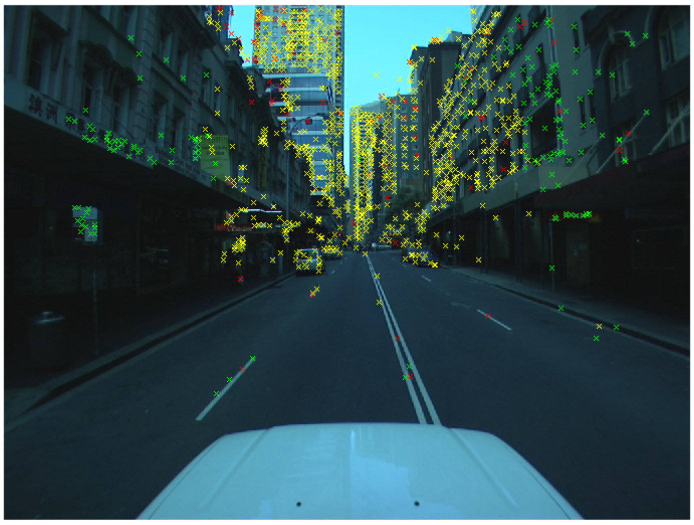
Selection of close by features

The final step is the approach to select those features that are close by and hence describe the local area around a reference place m_i best. In order to do that we make use of the fact that features being further away move less than close by ones in the image coordinate system when considering an image sequence. Thus, we consider the optical flow for all feature correspondences based on the images I_a and I_{a-1}. A vector is estimated from the locations of the features in the first frame to the one in the second frame. This vector has a magnitude d and a direction ϕ. Comparing the magnitudes of all correspondences, the relative distances to features with respect to our vehicles motion are obtained. This holds well for the straight motion along a street in urban canyons. Reference places are not picked during turns as, for instance, at intersections. However, there are problems involved. First, if the vehicle is not driving exactly in the middle of two building structures, the optical flow of the closer side is higher. That means it is likely that more features are selected from the closer side. Second, the scenario explained above only holds for features around static objects. Moving objects, such as other vehicles, cause a different optical flow in the image depending on their velocities with respect to the velocity of our vehicle.

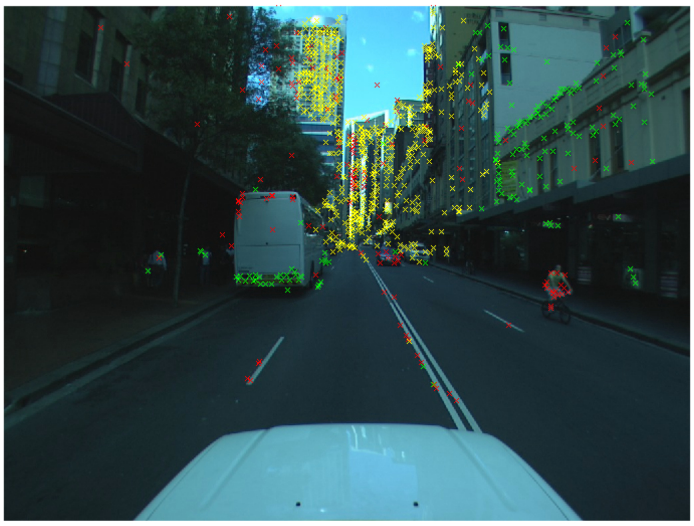
As the result of this step, we obtain $\tilde{N}$ feature correspondences sorted by the distance travelled over two images. Thus, we can choose the first $\hat{N}$ features of this set, where $\hat{N}$ denotes a constant number.

Examples

Figure 3.5a shows an example for the feature selection process described above. The features are assigned colours corresponding to their selection status. Remaining features after the matching with the previous image are plotted as red crosses. Features that successfully passed the geometric consistency check by RANSAC are plotted in yellow. Finally, the green crosses are the features saved for this reference place. Figure 3.5b illustrates another example. Here it is obvious that features around the bicyclist were detected as outliers. On the other hand, features around the parking bus are kept.



(a) *Example 1*



(b) *Example 2*

Figure 3.5: *Examples for the applied feature extraction and selection. Features remaining after matching are red, after RANSAC yellow. Green features are stored.*

3.3 Practical Considerations and Limitations

In this section the limitations of the proposed mapping procedure are summarized. In addition to that, details for practical applications are given.

As mentioned above, the position estimate based on odometry increases over time without reference measurements. In order to minimize the error in the distance measurement, it is important to calibrate the odometry before the mapping process. There are several sources of error for automotive platforms which can significantly deteriorate the precision of the odometry, as for example:

- tire inflation pressure
- varying load balance
- tire temperature differences
- tire slipping.

The first two examples can be addressed by the calibration. The others, in contrast, cannot be resolved by the calibration. It is highly demanding to account for this kind of errors. The calibration can be done by driving the vehicle a few times along a straight track whose distance is known. The conversion factor κ which translates wheel encoder pulses to wheel displacement and a standard deviation σ_κ can be estimated based on that.

The next problem addresses the flexibility of the map which is built as described previously. The places associated by SIFT features are only valid for the direction that the vehicle drove during the mapping process. If we drive along the same route but in the opposite direction, it is rather unlikely that places can be recognized. This is due to the fact that SIFT features are only partly invariant to affine distortions. The SIFT features usually do not match in case of major affine projections. However, this is not a general limitation. Morel et al. [19] proposed ASIFT as SIFT derivate which is fully invariant to affine distortions. The algorithm is computationally more expensive as a set of sample views of the initial image is simulated.

A further challenge is posed by the detection and rejection of features that describe dynamic objects. The RANSAC based approach is able to reject a multitude of these features, especially those that are close by. However, there might still be some undetected ones. The strategy of selecting close by features as introduced above might even impair since it privileges those with a higher magnitude in terms of their optical flow over two images. A vehicle on the opposite lane, for instance, moves towards our vehicle. Hence its optical flow is greater than those around static objects in the environment. A dense optical flow analysis could improve this. In addition to our approach the heading direction ϕ would be involved as well. In this way features could be tracked over multiple images using optical flow implementations as proposed by Kanade et al. [28] and Shi et al. [25]. An application to vehicle detection is given by Choi [7].

Another issue that is not covered by our feature selection regards the detection of features on the road surface as shown in Figure 3.5a. By the use of image processing algorithms like watershed[4] one could try to segment the road from the remainder of the image. Rejecting features detected on the road surface might seem obvious as lane markings are present throughout the entire environment. However, SIFT features base on local gradients, as explained in Section 2.6.1. Thus, lane markings can still possess distinctive features in their local appearance. For instance, Levinson et al. [17] build a map based on features extracted from road surfaces in urban environments. They demonstrated accurate localization based on that map. Although they apply different feature extraction techniques based on infrared images instead of SIFT, it still motivates to keep road surface features in our map.

Last, we will deal with the number of features that are kept. As mentioned previously, a fixed number of features $\hat{N}$ are stored for each reference place. This is done due to the selection of close by features. Distances to features are only estimated on a relative scale with respect to the vehicle's motion. The magnitudes depend on the actual distances and orientations of the features in the scene as well as the vehicle's velocity. Thus, we cannot simply take all features having travelled more than n pixels. In this way it might still happen that far away features are selected due to the fact that either less feature were observed or many were rejected before. If the minimum number of features $\hat{N}$ is not reached, the following images are considered until a maximum distance τ_{max} is exceeded and the map building process terminates with an error. This case is rather unlikely.

[4]More information about the watershed algorithm can be found in [29]

4 Localization

In this chapter, we present an approach for the localization of a vehicle using onboard odometry, more precisely wheel encoders and steering angle measurements, and a single perspective camera. Our localization system is given a feature map m, as discussed in chapter 3. As mentioned in the introduction, we address the problem of global localization. Thus, the system has to be able to estimate its position without an initial pose. This is the reason why we cannot use an extended Kalman filter as uni-modal probability distributions are not suitable for the global pose estimation [27, p. 194]. Instead we implement a particle filter which will be explained in the following. In section 2.2.1, we came up with a so-called motion model and an observation model. We referred to the motion model as the state transition probability $p(x'_t|x_t, u_t)$ propagating our system to state x'_t given the previous state x_t and the action, in this case a motion, u_t. Furthermore, the observation model is expressed as the likelihood $p(z_t|x_t)$ in terms of the estimation of the state belief $bel(x_t)$ (section 2.2.2). How both models are incorporated with a particle filter is described in section 2.2.4. The next sections deal with the implementation of both models. In addition to that, we will outline the applied resampling algorithm.

4.1 Motion Model

The location and orientation of the vehicle at a discrete time step t can be expressed as the pose x_t[1]:

$$x_t = (x, y, \theta)^T \tag{4.1}$$

At time step t, the motion u_t is carried out. It can be expressed as:

$$u_t = (v \, \omega)^T \tag{4.2}$$

[1]We only consider motion in 2D

The underlying drive train is the Ackerman steering, as described in section 2.1. Thus, we are given $v = v_c$ and $\omega = \omega_c$ for each motion u_t. The state transition from the previous state x_{t-1} to the state x'_t is defined as follow:

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x + v\Delta t cos(\theta + \omega\Delta t) \\ y + v\Delta t sin(\theta + \omega\Delta t) \\ \theta + \omega\Delta t \end{pmatrix} \tag{4.3}$$

The Equation 4.3 describes the exact motion after Δt units of time. However, the odometry is subject to noise which can occur due to a variety of sources as, for instance, wheel slipping on the ground, varying inflation pressures and the load balance of the vehicle. Also, the measurement of the steering angle is driven by noise. In order to incorporate this uncertainty in our motion model, we add Gaussian noise to the velocity v and the angular velocity ω. Thus, we can express the following:

$$\begin{pmatrix} \hat{v}' \\ \hat{\omega}' \end{pmatrix} = \begin{pmatrix} v + \epsilon_{\alpha_1|v|+\alpha_2|\omega|} \\ \omega + \epsilon_{\alpha_3|v|+\alpha_4|\omega|} \end{pmatrix}. \tag{4.4}$$

The variable ϵ_σ is the zero mean Gaussian noise with standard deviation σ. We set the standard deviation of the error according to the input velocity respectively angular velocity as proposed in [27, chapter 5]. The parameters $\alpha_1, ..., \alpha_4$ are platform specific and have to be set appropriately. For instance, the noise of the steering anle measurement of our vehicle can be significant. This is incorporated by adjusting the corresponding noise parameters α_2 and α_4. Thus, our final motion model can be defined as:

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x + \hat{v}\Delta t \cos(\theta + \hat{\omega}\Delta t) \\ y + \hat{v}\Delta t \sin(\theta + \hat{\omega}\Delta t) \\ \theta + \hat{\omega}\Delta t \end{pmatrix} \tag{4.5}$$

4.2 Observation Model

An observation z_t is made at the discrete time step t. This observation is applied to our probabilistic state estimation in form of the observation likelihood $p(z_t|x_t)$. As we are given a feature map m, this likelihood is extended to $p(z_t|m, x_t)$. In other words, this term expresses the probability of making the observation z_t given our current state x_t and the map m. To be more specific, observations in our case are camera images which are searched for visual features. We apply Scale Invariant Feature Transform (SIFT) as feature detector [18]. Thus, we observe a set of L features $z_t^{[1]}, ..., z_t^{[L]}$ at the time step t. In addition to that, each reference place of our map m contains a set of P local features. Thus, we try to associate observed features z_t with those stored for the reference places. This is the crux of our observation model and poses the basis for the assignment of the importance factor $w_t^{[k]}$ for the particle $x_t^{[k]}$ because of the following relationship:

$$w_t^{[k]} \propto p(z_t|m, x_t^{[k]}) \tag{4.6}$$

Different implementations in order to estimate the particles' weights are introduced in the following sections.

4.2.1 Simple Matching

In this section, we present an implementation for the observation model that works solely on similarity of SIFT descriptors. In this way, we compare all SIFT descriptors of the reference place m_i to the observed features z_t based on their Euclidian distances. Restating our matching function $match(m_i, z_t)$ of section 2.6.2, we get N_i as the number of corresponding features. Thus, we can say how likely it is that we make the observation z_t given we are at the reference place m_i. We model the likelihood $p(z_t|m_i)$ as a Gaussian:

$$p(z_t|m_i) = e^{\frac{-(N_i-\mu)^2}{\sigma_z^2}}, \tag{4.7}$$

whereas σ_z^2 denotes the variance measured on testing data sets and μ the fixed number of features for m_i as described in 3. In addition to that, we need a function expressing a distance metric between the particle $x_t^{[k]}$ and the reference place m_i. Thus, we define the following function:

$$f_{dist}(d) = \begin{cases} \exp(-\frac{(d-\mu_d)^2}{\sigma_d}) & (d > \mu_d) \\ 1 & (d \le \mu_d) \end{cases} \tag{4.8}$$

where μ_d and σ_d are empirically set according to the minimum distance of two reference places of m. The parameter d is the Euclidian distance of particle $x_t^{[k]}$ to the reference place m_i. This function returns rather optimistic values for particles being close to a reference place. This is firstly because we cannot properly estimate the distance to the reference place as we are only working in the space of appearance. That means a lower likelihood $p(z_t|m_i)$ for the place m_i does not necessarily occur due to a greater distance of the position where z_t was made and the reference place m_i. The reason for this can be occlusions or significant changes in illumination. As a result we observe less matching features and get a lower likelihood $p(z_t|m_i)$. In addition to that, we have an uncertainty about the actual position of the reference place m_i due to the map acquisition, as described in chapter 3. Using the likelihood $p(z_t|m_i)$ and the distance weighting function $f_{dist}(d)$, the observation likelihood for particle $x_t^{[k]}$ can be expressed as:

$$p(z_t|m, x_t^{[k]}) = \sum_{i=1}^{M} p(z_t|m_i) \cdot f_{dist}(\|x_t^{[k]} - X_{m_i}\|) \tag{4.9}$$

The weight $w_t^{[k]}$ is assigned proportional to this likelihood. The normalization of the weights is carried out accordingly. Note, that even though the particle's state actually expresses a pose, only the location in (x, y) is used for the calculation of the Euclidian distance to the reference place m_i with the location X_{m_i}. As a result, we obtain the observation likelihood for the particle $x_t^{[k]}$ as the sum of the likelihoods of all M reference places weighted by their distances. Estimating the observation likelihood in this way, surely obtains high confidence as the entire state space is considered for each particle. Using this estimate, however, is computationally very expensive as a lot of features with high-dimensional descriptor vectors are considered. Especially in the case, when the particle filter converged around the true posterior, the consideration of reference places being far away from the belief $bel(x_t)$, is not necessary. Thus, the estimate of the observation likelihood is simplified as follow:

$$\tilde{M} = \{m_i \in M | \beta > \|x_t^{[k]} - X_{m_i}\|\} \tag{4.10}$$

$$p(z_t|m, x_t^{[k]}) = \sum_{m_i \in \tilde{M}} p(z_t|m_i) \cdot f_{dist}(\|x_t^{[k]} - X_{m_i}\|). \tag{4.11}$$

We use $\tilde{M}$ in order to express a set of all locations having a distance less than β to the particle $x_t^{[k]}$. In this way, the computational requirements are significantly reduced while the result is not deteriorated, providing β is set appropriately.

4.2.2 Histogram based Approach

So far, we discussed an implementation for the observation model using the classical approach of matching SIFT descriptors. The crucial point poses the determination of the threshold. Even though the use of further geometric constraints allows to increase this threshold, the matching simply says whether a descriptor correspondence is valid based on their similarity. However, there is no information given about the strength of similarity besides the fact whether it falls below the threshold or not. Thus, we want to introduce a probabilistic approach to this problem which is motivated by Bennewitz et al. [3]. The basic idea is to replace the matching function used so far. In this way we define $f_i^1, ..., f_i^P$ as the set of descriptor vectors associated with the features stored for the reference place m_i. Furthermore, the descriptor vectors associated with z_t are defined as $f'^1_t, ..., f'^L_t$. The likelihood that the vectors $f_i^{[p]}$ and $f'^{[l]}_t$ describe the same feature is estimated as:

$$p(f_i^{[p]} \equiv f'^{[l]}_t) = \exp(-\frac{\|f_i^{[p]} - f'^{[l]}_t\|}{2 \cdot \sigma_1^2}), \tag{4.12}$$

where σ_1 denotes the variance of the Gaussian. For each feature of the place m_i the similarity likelihoods to all features observed are estimated according to Equation 4.12. Thus, $P \times L$ likelihoods are obtained. Next, we select for each feature $f_i^{[p]}$ of the reference place the maximum similarity likelihood $max(p(f_i^{[p]} \equiv f'^{[1]}_t), ..., p(f_i^{[p]} \equiv f'^{[L]}_t))$. We denote the combination of the stored feature $f_i^{[p]}$ with its most similar observed feature $f'^{[l]}_t$ as $(f_i^{[p]}, f'^{[l]}_t)$. The set C contains all P combinations. These combinations are used in order to estimate the observation likelihoods for the particles.

As we are using monocular vision the depth to observed features is unknown. Hence only the bearings can be used. In this way one could compare the pixel displacements of stored features and observed features. However, calculating the Euclidian distances is crucial as the magnitudes are significantly influenced by the vertical displacements. This becomes apparent when a vehicle just drives in a different lane than during the mapping process. This magnitude would be quite large even though the vehicle is almost at the same position if we consider the street as a one-dimensional line. The magnitude could be the same for a vehicle being further away but driving in the same lane as during the mapping process. Thus, only the horizontal pixel displacements are considered to compute the observation likelihoods. To be more precisely, a distribution over the horizontal pixel displacements of the features stored for the reference place m_i and the observed features is estimated. Therefore, a histogram is computed with each bin representing a range of displacements. For each feature $f_i^{[p]}$ of the reference place m_i we have a likelihood and a pixel displacement with its associated most similar observed feature. The similarity likelihoods are added to the bin with the corresponding pixel displacement.

Hence each bin b contains a sum $h(b)$ and is bordered by the displacement values $\alpha^-(b)$ and $\alpha^+(b)$. The value $h(b)$ of bin b can be defined as

$$h(b) = \sum_{\{(f_i^{[p]}, f'^{[l]}_t) \in C \mid \alpha^-(b) \leq \|\alpha(f_i^{[p]}) - \alpha(f'^{[l]}_t)\| < \alpha^+(b)\}} p(f_i^{[p]} \equiv f'^{[l]}_t). \tag{4.13}$$

Here the function $\alpha(\cdot)$ expresses the horizontal pixel coordinate of a feature. Consequently, we obtain the distribution about the displacements. The final value should rather depend on the similarities of features. However, the closer a bin to zero, the closer are the pixel displacements of its associated feature combinations. The density extracted from that distribution depends on the similarity of stored features and observed features. However, a high similarity should be weighted less if the pixel displacement is high. First, this minimizes the influence of wrong matches since the positions of the associated features often deviate significantly. Wrong matches in this case means feature correspondences that are assigned a high likelihood due to their similarity. However, they belong to different features. Second, it roughly evaluates the relative distance of the stored features and observed features. Let us imagine we are approaching a place which poses a reference place in our map. The closer we get to the original position the less the horizontal pixel displacement of corresponding features. However, one cannot relate this to an actual metric saying the vehicle has a specific distance to a reference place because the stored features have different positions in the scene.

In order to estimate the observation likelihood a density has to be extracted from the distribution determined above. The value of each bin is weighted by a zero mean Gaussian according to the pixel displacements expressed by its boundaries $\alpha^+(b)$ and $\alpha^-(b)$. This can be expressed as:

$$p(z_t|m_i) = \sum_b h(b) \cdot \exp\left(-\frac{1}{2\sigma_2^2} \cdot \left[\frac{\alpha^+(b) + \alpha^-(b)}{2}\right]^2 \right) \tag{4.14}$$

The observation likelihood $p(z_t|m_i, x_t^{[k]})$ for the particle $x_t^{[k]}$ is assigned according to Equation 4.9. The simplification expressed by 4.10 also holds for this observation model.

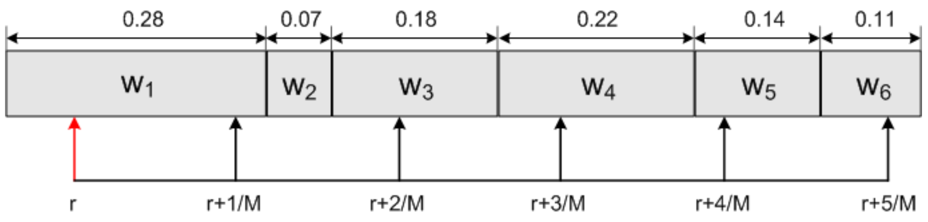


Figure 4.1: *Low variance sampling*

4.3 Resampling

The resampling step poses one of the most crucial steps in a particle filter framework. The main goal is to achieve a higher probability density in areas of high interest which is determined by the particles' weights. However, due to the random sampling nature of the particle filter, we have to deal with the variability of the samples. This is called the sampling variance [27]. For instance, if exactly the same action is carried out simultaneously on two platforms with equal properties, we get two different kernel densities. The variance in these densities decreases with the number of particles representing the state space. Each resampling step normally reduces the variance of the particle set itself, but in contrast the variance of the particle set in terms of the estimation of true belief increases [27]. This becomes apparent, when, for example, a vehicle stops in front of a traffic light. If we keep applying observations and resampling, one particle would possess the entire probability mass after a while. The diversity of the particle filter is lost. Thus, the resampling is deferred when the vehicle is not moving. Also, the observations during this time are rejected. Bennewitz et al. [3], for example, estimate a number of effective particles and perform a resampling only when this number drops below a certain threshold. However, we could not observe an advantage by applying this.

Another way of reducing the effect explained above is to apply low variance sampling [27]. In contrast to the resampling strategy introduced in Section 2.2.4, we use only one random number r and select the remaining samples based on this. Algorithm 3 shows the low variance resampling which is explained in the following using the example illustrated by Figure 4.1. Each box of Figure 4.1 denotes a weight w_i with a value as shown on top of it. Before the resampling step, the set of the size $M = 6$ contains the particles $i = \{1, 2, 3, 4, 5, 6\}$. Firstly, a random number within the interval $[0; M^{-1}]$ is chosen (line 3). The variable c is initialized with the first particle's weight, the index i is set to one. U is initialized with the random number r which points to our first weight. As U is less than 0.28, the first particle is added to the new set. U is incremented by M^{-1} and still points on w_1. Hence, the first particle is added again to the new particle set. Next, U is incremented by M^{-1} , thus c drops below U. Consequently, c points to the end of w_2 now. As U is still greater than c, c is shifted to the end of w_3 and $i = 3$. Now, U is less than c and the particle 3 is added to the new particle set. At the end of this procedure, we obtain the set $i = 1, 1, 3, 4, 5, 6$.

The advantages of low variance sampling are obvious. First, it works more systematically as it cycles through all particles. In addition to that it has a complexity of $O(M)$ [27]. There is a variety of other resampling algorithms which can be found, for instance, in [27, chapter 4] and [10].

Algorithm 3 Low Variance Sampling

```
1:  function LowVarianceSampling($\{x_t^{[1]}, ..., x_t^{[K]}\}, \{w_t^{[1]}, ..., w_t^{[K]}\}$)
2:      $r = rand(0; M^{-1})$                                   ▷ Initialization
3:      $c = w_t^{[1]}$
4:      $i = 1$
5:      for k=1 to K do
6:          $U = r + (m - 1) \cdot M^{-1}$
7:          while $U > c$ do
8:              $i = i + 1$
9:              $c = c + w_t^{[i]}$
10:         end while
11:         $x_t^{[k]} := \bar{x}_t^{[i]}$                         ▷ add particle with index i
12:     end for
13:     return $\{\bar{x}_t^{[1]}, ..., \bar{x}_t^{[K]}\}$
14: end function
```

5 Experimentation

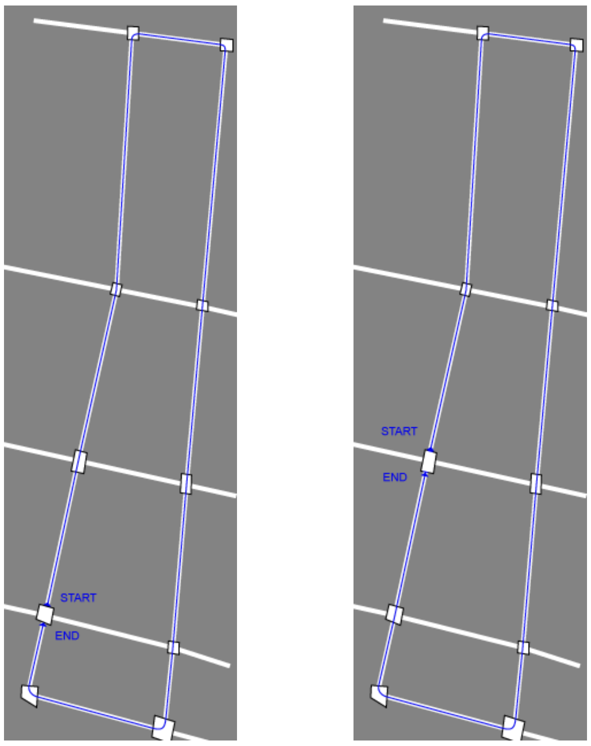
In order to evaluate our system, experiments on real roads were carried out. The majority of the algorithms are implemented in Matlab and not optimised for real-time application. Thus, the sensor data was logged during the experiment and processed afterwards. As testing environment a loop in a dense urban environment of the Central Business District of Sydney, Australia was selected. In this area there are many high building structures present. The streets are mostly very narrow. The operation of GPS in this area is unreliable as the perceptible satellites are arranged very tight. Only a few street segments and intersections provide a wider field and hence a better satellite constellation. In addition to that, the experimentation environment is characterized by a lot of dynamic objects as pedestrians and other vehicles. The vehicle is driven in a way that is typical of humans. That means that the vehicle was not purposely driven at slower speeds. In the following the mapping as well as the localization implementations as explained in the previous chapters are presented. First, it is shown how the map was built. Afterwards, the localization based on that map is carried out in two different experiments. Here, the focus is the general evaluation as well as the comparison of different implementations.

5.1 Map building

As the precision of the odometry is of high importance for the mapping process, a calibration was carried out beforehand. Afterwards the vehicle was driven one loop on the specified route. The map of the environment was built as explained in Chapter 3. Street segments were defined based on intersection lines. Every 10 metres, the closest frame based on time stamps was selected as reference frame and hence a reference place was defined based on this. The minimum distance between reference places was set to 10 metres and the maximum distance to 20 metres. For each reference place we stored 250 SIFT features.

5.2 Experiment 1

The first experiment for the localization was carried out right after the mapping process. That means the sensor data of the first loop was used to build the map, the following one for the localization. This poses the best preconditions in terms of similarity as illumination has not perceptibly changed in the meantime. Both implementations for the observation model discussed in Section 4.2 are applied. The ground truth is given in terms of start and end of the trajectory and the directions (see Figure 5.1a). It is not based on measurements and hence manually set.



(a) *Experiment 1* (b) *Experiment 2/3*

Figure 5.1: *Ground truth trajectories*

The loop start and end are determined similarly to the mapping process. Those images that are captured exactly at intersection lines are manually associated with corresponding reference places. A set of $K = 500$ particles was used to estimate the position. The vehicle's pose x_{bel} is estimated as the weighted mean over the particles:

$$x_{bel} = \sum_{k=1}^{K} w^{[k]} x^{[k]} \tag{5.1}$$

The uncertainty in the estimate is given in terms of the variance over the particles weighted mean:

$$Var(x_{bel}) = \sum_{k=1}^{K} \bar{w}^{[k]}(x^{[k]})^2 - \left(\sum_{k=1}^{K} w^{[k]}x^{[k]}\right)^2. \tag{5.2}$$

The results for both implementations are shown in Figure 5.2 In addition to that the uncertainties are shown.

5.3 Experiment 2

A second experiment was carried out. Basis for the localization was the same map as in the previous experiment. Hence localization and mapping base on data sets of completely different times as about two months had elapsed in-between. In this way, it is highly unlikely that objects as parking vehicles, whose appearances were saved for reference places, are recognized at the same location. In addition to that, we have slightly different lightning conditions. Again, a set of 500 particles was used. The ground truth is shown in Figure 5.1b results are shown in Figure 5.3.

5.4 Experiment 3

The goal of the final experiment is the evaluation of the feature selection as discussed in Section 3.2. Hence we apply the simple matching approach using the map with selected features as for the previous experiments. Second, we apply the same approach, but using all features associated with the reference places. The observation model as defined in Equation 4.7 is slightly changed as the number of features for each reference place varies:

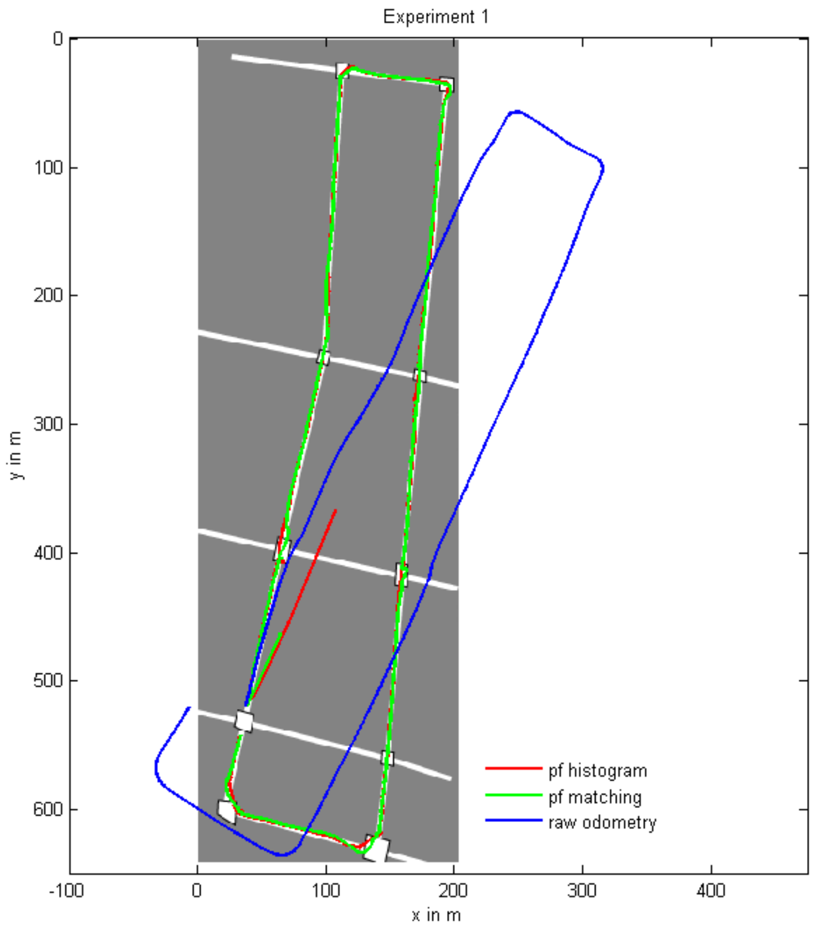
$$p(z_t|m_i) = e^{\frac{-(N_i-P_i)^2}{\sigma_z^2}}, \tag{5.3}$$

where P_i denotes the number of features of the reference place m_i. The underlying data set is the same as in the second experiment (see Section 5.3). Thus, ground truth is given by Figure 5.1b as well. The results are shown in Figure 5.4.

5.5 Summary

The results of the experiments can be summarized as follows. First, it can be said that the global localization using a single camera, odometry and a prior map is possible. The second experiment proved that the localization also worked at a completely different time. Nevertheless the experimental results vary from the expectations. The histogram and the matching based derivates performed similarly in both experiments. Due to the change in illumination in the second experiment the histogram based implementation was expected to perform better than the matching based derivate as latter might reject many correspondences that fall above the threshold. However, this could not be observed. Both implementations perform similiarly in these experiments. The histogram approach shows some minor drifts. An uncertainty at intersections especially when turning becomes apparent for both derivates, particularly at the lower intersections on the map. This is because these intersections provide a wider field of view in contrast to the others. Thus, the similarities of reference places close to these intersections is higher.

The third experiment dealed with the comparison of the localization based on a map built using feature selection and the same map but using all features that were observed at a reference place. The trajectories as well as the uncertainties do not show significant differences. However, a major increase in the processing time was recorded because there are typically about 2000 features for each reference place. The case that, for instance, a dynamic object was mapped at one reference place and appeared at a different one might not have occurred. In addition to that, dynamic objects probably confuse rather in the initialization step respectively during phases of higher uncertainties. Once the particle filter converged close to the true position, the influence of these observations is minimized. All in all, apart from processing time there were no particular advantages of the feature selection in combination with the particle filter in this experiment.



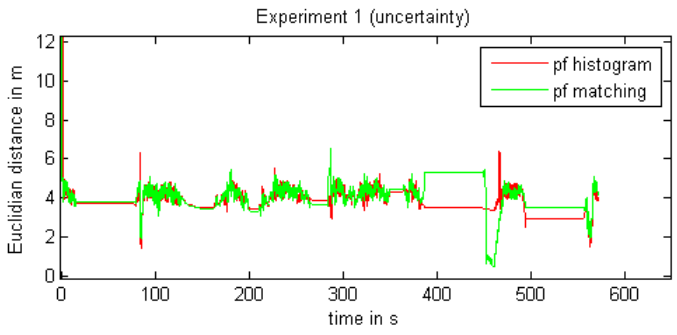
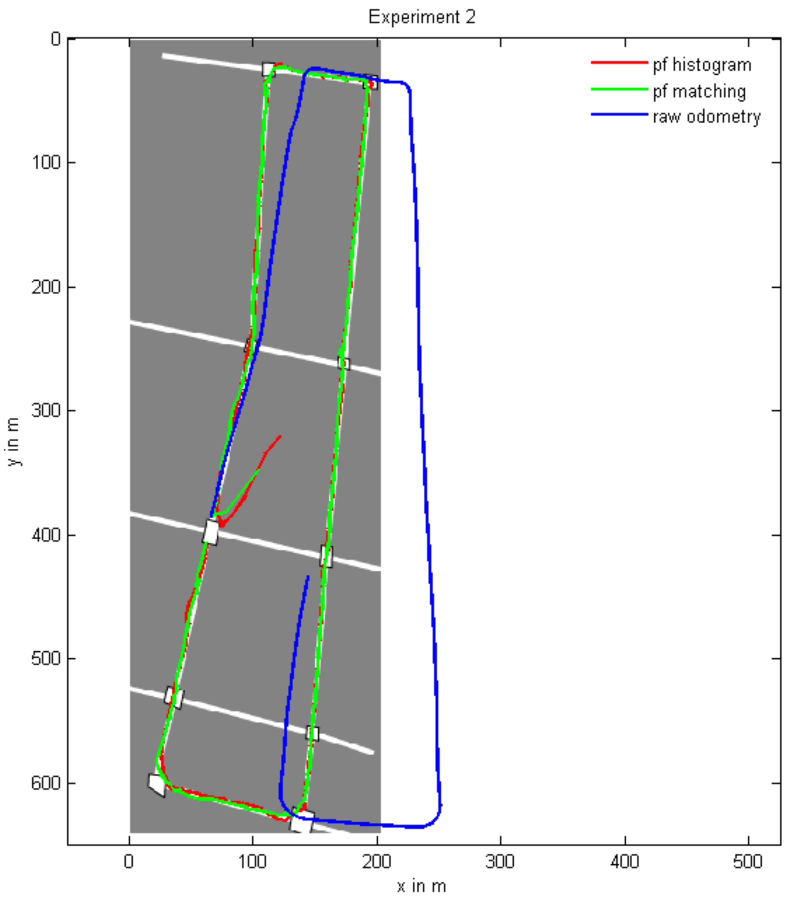


Figure 5.2: *Results of experiment 1*



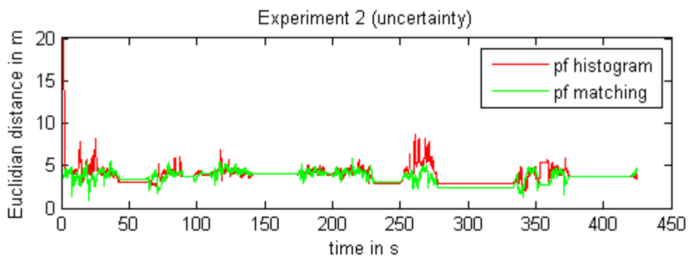
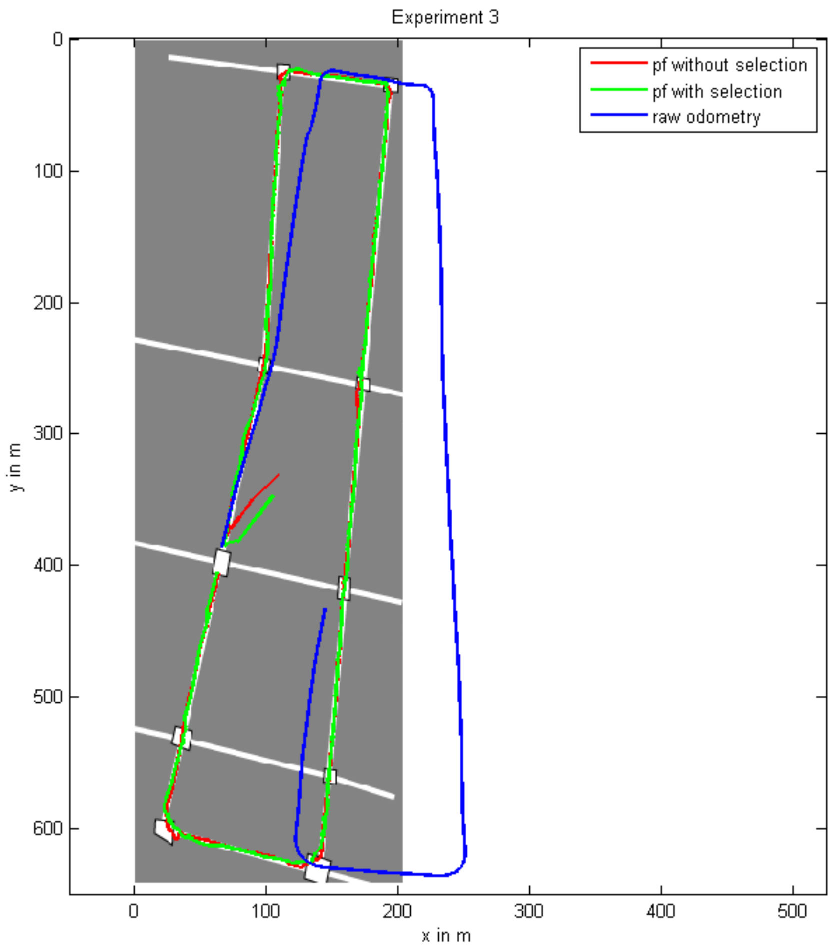


Figure 5.3: *Results of experiment 2*



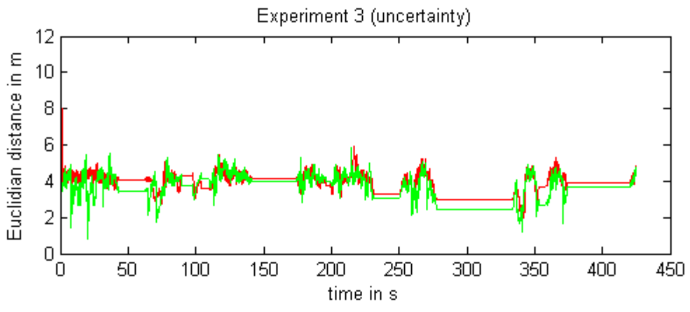


Figure 5.4: *Results of experiment 3*

6 Discussion

6.1 Conclusion

In this thesis, we presented an approach for the localization of autonomous ground vehicles in dense urban environments using monocular vision and odometry. It was explained how a hybrid map containing reference places of the environment as well as metric information can be built. In particular, we used satellite images to obtain information about intersections and courses of street segments. The map was supplemented with GIS data which enabled an appropriate scaling. Based on odometry and camera images we defined reference places on this map. Approaches for a feature selection in order to obtain distinctive reference places and a compact map representation were shown.

The global localization was addressed in terms of a place recognition problem using a probabilistic framework. A particle filter was implemented to estimate the vehicle's pose given motions based on odometry and observations in form of SIFT features. Observed features were associated with those stored for reference places of our map. We presented two different approaches implementing this in respect of the observation model. One evaluates the similarity of observed and stored features by a distance threshold. The other one tackles the similarity measurement in a probabilistic manner while simultaneously incorporating geometric constraints.

The presented experimental results show that our appearance based approach can successfully localize a vehicle given a map with reference places.

6.2 Limitation

Even though the localization based on the presented approach is possible there are limitations. As we are working in the space of appearance a high precision estimate on a centimetre scale is not possible. However, it enables a reliable localization on street level. The position estimate is highly reliant on the accuracy of the map, particularly the locations of the reference places. Moreover, it is influenced by the distance between reference places.

Second, SIFT features are only invariant to a certain amount of illumination change. Thus, the localization throughout completely different times of day, such as day and night, is crucial. Nevertheless one could build maps containing reference places and associated features for a variety of illumination conditions.

6.3 Future work

Instead of assigning locations to reference places based on odometry distance measurements, a Simultaneous Localization and Mapping (SLAM) approach should be applied. This would decrease the uncertainty of these locations. Our experimental trajectories were about 1.5 km which poses a challenge to visual SLAM algorithms as the complexity is high due to the number of features. Thus, an algorithm as proposed by Huang et al. [14] working with local sub-maps that are connected to a global map is advantageous. Only features in the boundaries of two local sub-maps are kept for data association. We applied this algorithm using the same data set as for our second experiment (see Section 5.3). Due to the scale ambiguity of monocular vision, the final trajectory is up to one global scale. The scaling was carried out using odometry between two close-by poses. The result is shown in Figure 6.1. It is apparent that there is a major drift in the upper part of the trajectory. This might have occurred due to the presence of a multitude of dynamic objects as particularly this region is very busy. Further investigation in the rejection of features around dynamic objects could potentially help. Alternatively the consideration of smaller trajectories might help.

Next, there are possible extensions for our approach regarding the representation of reference places. Instead of saving high-dimensional SIFT descriptors visual vocabulary trees could be applied. The basic idea is to represent features as visual words. A vocabulary tree is built based on the visual words. First, a query word is compared to the leaf nodes. From the closest leaf node the tree is recursively traversed until the closest visual words is found. This node can, for instance, link to a specific reference place. This is similar to text recognition algorithms and well suited for a large number of reference places. An approach for large city scale location recognition is given by Schindler et al. [24].

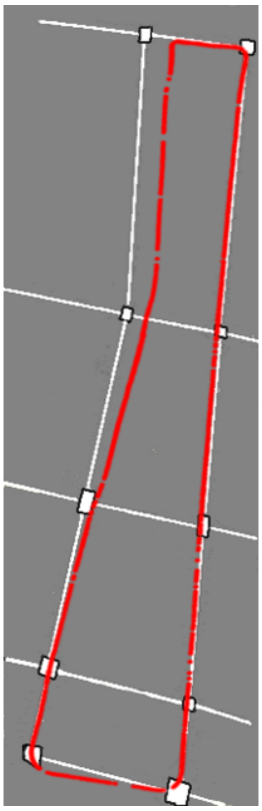


Figure 6.1: *Scaled SLAM trajectory plotted on a prior map*

Even if visual vocabulary approaches are applied, it makes sense to use the GPS signal as prior for the initialization of the particle filter. In this way particles are only initialized in the region covered by the uncertainty of the GPS instead of the entire state space. In order to recover from localization errors, it is recommended to continuously sample a set of particles with random states [26]. However, this involves further changes as the belief cannot be estimated as the weighted mean over all particles as presented in this thesis. More complex strategies as kernel density estimation or density trees could be implemented [27, p. 104 et seq.].

The next extension addresses the odometry. As it became apparent in the experiments, the vehicle's steering angle measurement is subject to greater noise, particularly when turning at intersections. Further investigation in the use of the onboard available inertial measurement unit in order to achieve a better prediction for the localization is intended.

Finally, the localization approach should be optimised for the application in real-time. This is possible by the use of vocabulary trees supported by the GPS prior information during the initialization as mentioned above. Also, the extraction of SIFT features can be carried out in real-time by the use of parallel hardware architectures. Wu et al. [30] proposed a SIFT implementation based on graphic processor units (GPU) which enables the operation at about 15 Hz for images with a resolution of 1024×768 pixels. Chang et al. [6] presented an approach for the implementation of SIFT on Field Programmable Gate Arrays (FPGA) which significantly accelerates the feature extraction while simultaneously working more economically in terms of power consumption compared to graphic processor units.

A Appendix

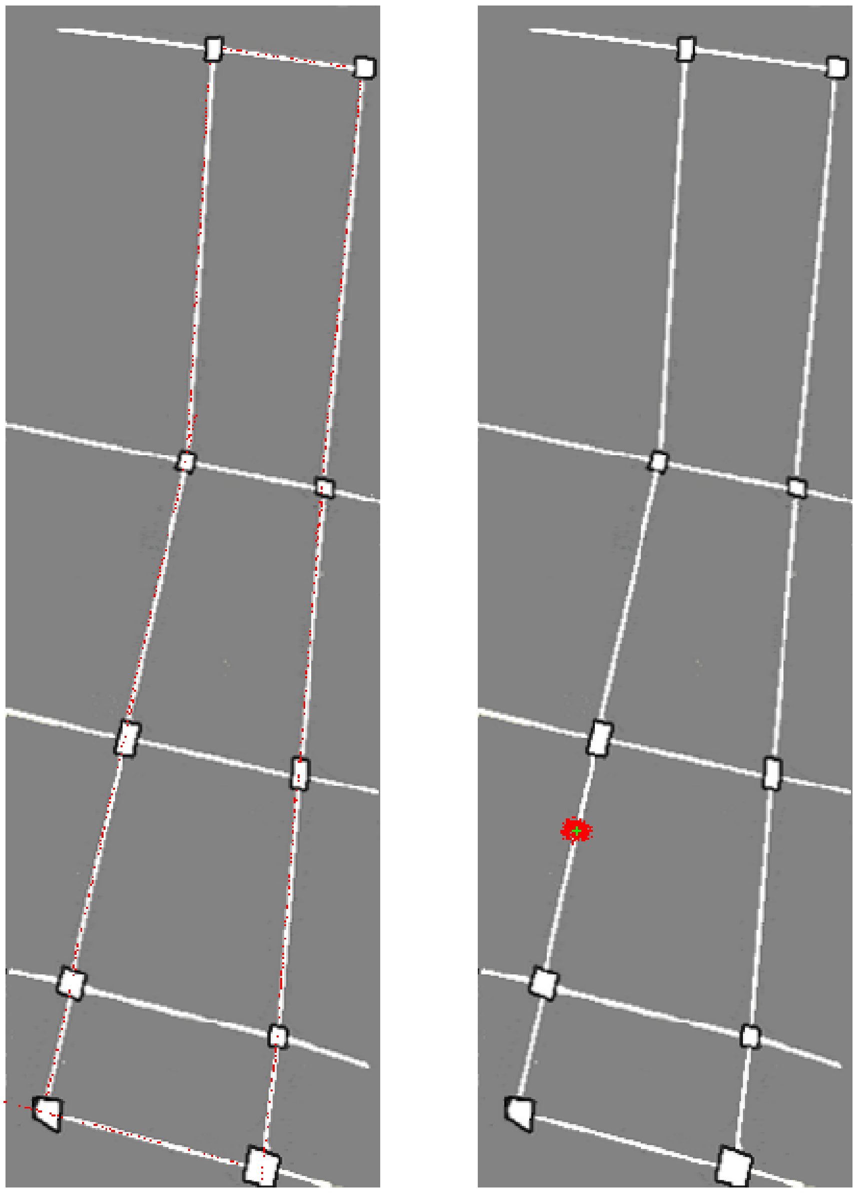
A.1 Discrete Convolution in 2D

Providing f and g are two functions from $(X, Y) \subset \mathbb{Z}^2$ to $\mathbb{R}$, the convolution of f and g is defined as follows [26]:

$$(f \otimes g)(x, y) := \sum_{u \in X} \sum v \in Y f(u, v) \cdot g(x - u, y - v) \tag{A.1}$$

Convolutions are also used if the domains of the functions are dissimilar. The functions are extended with zeros in this case.

A.2 Particle Filter Implementation



(a) *Initialization* (b) *Particle filter converged*

Figure A.1: *Screenshots of the particle filter implementation with particles (red) and estimated belief (green cross)*

A.3 Theses

1. The vehicle localization in dense urban environments can be realized using approaches that work with the appearance of visual features.
2. SIFT features are well suited for the vision based global localization.
3. Feature selection helps distinguishing places with high similarity in place recognition systems and enables a more compact representation.
4. A RANSAC based approach to the detection of outliers can reject a multitude of dynamic objects during the map acquisition.
5. A feature selection for place representations in combination with a particle filter could not proved to be beneficial.
6. The probabilistic approach to the observation model using sparse geometric constraints did not show advantages compared to the simple matching.

Bibliography

[1] M. S. Arulampalam, S. Maskell, and N. Gordon. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2002.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer Vision (ECCV 2006)*, pages 404–417, 2006.

[3] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric localization with scale-invariant visual features using a single perspective camera. In *EUROS*, pages 195–209, 2006.

[4] O. Booij and Z. Zivkovic. The planar two point algorithm. Technical report, Intelligent Systems Laboratory Amsterdam, University of Amsterdam, September 2005.

[5] C. Boor. *A practical guide to splines*. Applied mathematical sciences. Springer, 2001.

[6] L. Chang and J. Hernández-Palancar. A hardware architecture for sift candidate keypoints detection. In E. Bayro-Corrochano and J.-O. Eklundh, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, volume 5856 of *Lecture Notes in Computer Science*, pages 95–102. Springer Berlin / Heidelberg, 2009.

[7] J. Choi. Realtime on-road vehicle detection with optical flows and haar-like feature detector. Technical report, University of Illinois at Urbana-Champaign, 2006.

[8] I. Cox. Blanche: An experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7:193–204, 1991.

[9] M. Cummins and P. Newman. FAB-MAP: Probabilistic Localization and Map-

ping in the Space of Appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[10] A. Doucet, N. Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice.* Statistics for engineering and information science. Springer, 2001.

[11] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[12] T. Goedemé, T. Tuytelaars, and L. J. V. Gool. Visual topological map building in self-similar environments. In *ICINCO-RA*, pages 3–9, 2006.

[13] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* Cambridge University Press, 2003.

[14] S. Huang, Z. Wang, G. Dissanayake, and U. Frese. Iterated d-slam map joining: evaluating its performance in terms of consistency, accuracy and efficiency. *Autonomous Robots*, 27:409–429, 2009. 10.1007/s10514-009-9153-8.

[15] L. F. J. Borenstein, B. Everett. Where am i? sensors and methods for mobile robot positioning. Technical report, University of Michigan, March 1996.

[16] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[17] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[18] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal Computer Vision*, 60:91–110, November 2004.

[19] J. M. Morel and G. Yu. ASIFT: A New Framework for Fully Affine Invariant Image Comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[20] J. Nieto, J. E. Guivant, and E. M. Nebot. The hybrid metric maps (hymms): a novel map representation for denseslam. In *ICRA*, pages 391–396, 2004.

[21] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26:756–777, June 2004.

[22] W. Norris. *Modern Steam Road Wagons.* Audubon Press & Christian Book Service, 2007.

[23] D. Scaramuzza, F. Fraundorfer, and R. Siegwart. Real-time monocular visual

odometry for on-road vehicles with 1-point ransac. In *Proc. of The IEEE International Conference on Robotics and Automation (ICRA)*, May 2009.

[24] G. Schindler, M. Brown, and R. Szeliski. City-Scale Location Recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–7, 2007.

[25] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994.

[26] H. M. Strasdat. Localization and mapping using a single-perspective camera. Master's thesis, University of Freiburg, July 2007.

[27] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics.* Intelligent robotics and autonomous agents. MIT Press, 2005.

[28] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, Carnegie Mellon University, April 1991.

[29] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE PAMI, 1991*, 13(6):583–598, 1991.

[30] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). http://cs.unc.edu/ ccwu/siftgpu, 2007.

[31] G. Xu. *GPS: theory, algorithms, and applications.* Springer, 2007.