FastSLAM with Stereo Vision

Wikus Brink
Electronic Systems Lab
Electrical and Electronic Engineering
Stellenbosch University
Email: wikusbrink@ieee.org

Corné E. van Daalen
Electronic Systems Lab
Electrical and Electronic Engineering
Stellenbosch University
Email: cvdaalen@sun.ac.za

Willie Brink
Applied Mathematics
Department of Mathematical Sciences
Stellenbosch University
Email: wbrink@sun.ac.za

***Abstract*—We consider the problem of performing simultaneous localization and mapping (SLAM) with a stereo vision sensor, where image features are matched and triangulated for use as landmarks. We explain how we obtain landmark measurements from image features, and describe them with a Gaussian noise model for use with a Rao-Blackwellized particle filter-based SLAM algorithm called FastSLAM. This algorithm uses particles to describe uncertainty in robot pose, and Gaussian distributions to describe landmark position estimates. Simulation and experimental results indicate that FastSLAM is well suited for vision-based SLAM, because of an inherent robustness to landmark mismatches, and we achieve accuracies that are comparable to other state-of-the-art systems.**

I. Introduction

Simultaneous localization and mapping (SLAM) is a rapidly growing part of the autonomous navigation field. SLAM attempts to solve the problem of estimating a mobile robot's position in an unknown environment while building a map of the environment at the same time. This is a challenging problem since an accurate map is necessary for localization and accurate localization is necessary for mapping.

Most SLAM algorithms use a probabilistic landmark-based map rather than a dense map. If landmarks in the map can be measured, relative to the robot, and tracked over time the pose of the robot and the locations of the landmarks can be estimated in an optimal manner.

Initial implementations made use of the extended Kalman filter (EKF), but displayed several shortcomings such as quadratic complexity and sensitivity to incorrect feature tracking [1] [2]. The particle filter can be used to overcome these limitations. However, because of the high dimensionality of the problem the particle filter cannot be used directly. Instead, the Rao-Blackwellized particle filter [3] is used. This filter estimates some states with particles and others with EKFs. In the case of SLAM particles are used for the pose of the robot and an EKF for each landmark. This method is called FastSLAM and has shown promising results in the literature [4] [5].

Stereo vision is an attractive sensor to use with SLAM as it can provide a large amount of 3D information at every time step. Extracting that information reliably can, however, be challenging. Powerful algorithms such as SIFT [6] or SURF [7] have been used to solve this problem by extracting salient features from images. These algorithms can be employed to track features over multiple images so that landmarks for SLAM can be identified.

In this paper we attempt to solve the 2D SLAM problem by using FastSLAM and image features (the 3D extension is conceptually the same). We begin with a brief description of how we obtain measurements of landmarks with a Gaussian noise model. A detailed description of the FastSLAM algorithm is given, followed by some simulations where we compare FastSLAM with the popular EKF SLAM algorithm [2]. We provide experimental results from our system on an outdoor dataset and measure accuracy against differential GPS ground truth.

II. Image features and stereo geometry

In this section we discuss a method of finding features in images, triangulating these features for use as landmarks and approximating the noise associated with each measurement of a landmark. This characterization of the stereo vision sensor is important for accurate optimal estimation. Since this section is similar to previous work, the explanation will be brief. For a more in depth discussion refer to [8] and [9].

A. Feature detection and matching

In order to identify landmarks we opt for one of two popular feature detection algorithms: the scale-invariant feature transform (SIFT) [6] or speeded-up robust features (SURF) [7]. Note that since we perform SLAM in 2D we discard the vertical coordinates of image features.

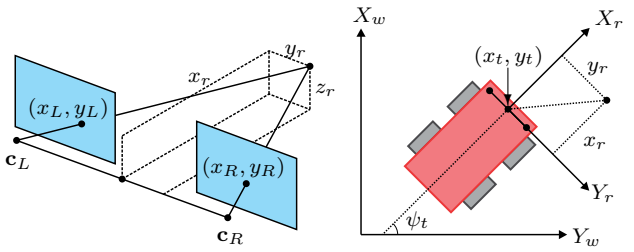
At every time step we search for feature matches in a synchronized pair of rectified stereo images. We model each match as a measurement with Gaussian noise:

$$\mathbf{z}_{\text{im}} = \begin{bmatrix} x_L \\ x_R \end{bmatrix} + \mathcal{N}(\mathbf{0}, \mathbf{N}_t), \tag{1}$$

where x_L and x_R are the image coordinates of the feature in the left and right images. By $\mathcal{N}(\mathbf{0}, \mathbf{N}_t)$ we mean a sample drawn from the normal distribution with zero mean and covariance matrix $\mathbf{N}_t$ (the same notation is used throughout the rest of this paper). We describe the noise covariance in Equation 1 by

$$\mathbf{N}_t = \begin{bmatrix} \sigma_{x_L}^2 & 0 \\ 0 & \sigma_{x_R}^2 \end{bmatrix}, \tag{2}$$

with σ_{x_L} and σ_{x_R} the standard deviations in pixels of the match measurement, which we obtain through testing.



(a) camera geometry (b) robot geometry

Fig. 1. The geometry of our system.

We can then match the descriptors of a new measurement with the descriptors of features already found at previous time steps, to arrive at putative landmark correspondences.

B. Stereo geometry of calibrated images

Now that we have stereo image features that can be tracked over time, we convert them into 2D landmarks.

Figure 1(a) depicts the geometry of a pair of stereo cameras with camera centres at $\mathbf{c}_L$ and $\mathbf{c}_R$, where the image planes have been rectified, and a landmark $\begin{bmatrix} x_r & y_r & z_r \end{bmatrix}^T$ observed at image coordinates (x_L, y_L) in the left image and (x_R, y_R) in the right image. As mentioned we are working in 2D, so the features are effectively projected onto the $X_r - Y_r$ plane.

With the geometry of the stereo camera pair, the landmark location in metres can be calculated in robot coordinates as

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} \frac{fb}{x_L - x_R} \\ \frac{(x_L - p_x)b}{x_L - x_R} - \frac{b}{2} \end{bmatrix} + \mathcal{N}(\mathbf{0}, \mathbf{Q}_t), \tag{3}$$

where b is the baseline (distance between $\mathbf{c}_L$ and $\mathbf{c}_R$), f the focal length and p_x and p_y the x- and y-offset of the principal point, all obtained from an offline calibration process. $\mathbf{Q}_t$ is the noise covariance matrix of the measurement.

Note that we differentiate between robot coordinates (subscript r) and world coordinates (subscript w) as indicated in Figure 1(b), where x_t, y_t and ψ_t are the robot's position and orientation in world coordinates at time t.

We know that a transformation from $\mathbf{N}_t$ to $\mathbf{Q}_t$ is possible if we have a linear system and, since Equation 3 is not linear, we use a first order Taylor approximation to find the transformation matrix

$$\mathbf{W}_t = \begin{bmatrix} \frac{\partial x_r}{\partial x_L} & \frac{\partial x_r}{\partial x_R} \\ \frac{\partial y_r}{\partial x_L} & \frac{\partial y_r}{\partial x_R} \end{bmatrix}. \tag{4}$$

It then follows that $\mathbf{Q}_t$ can be approximated as

$$\mathbf{Q}_t = \mathbf{W}_t \mathbf{N}_t \mathbf{W}_t^T. \tag{5}$$

This approximation is performed to maintain a Gaussian noise model, which is necessary for FastSLAM. We use this noise model and the triangulated locations of landmarks to find outliers in putative correspondences between new measurements and those already in the map, according to the RANSAC-based probabilistic method discussed in [9].

From Figure 1(b) we see that the robot pose can be described with the state vector

$$\mathbf{x}_t = \begin{bmatrix} x_t \\ y_t \\ \psi_t \end{bmatrix}, \tag{6}$$

with x_t and y_t the location of the robot and ψ_t its orientation. We define the rotation matrix

$$\mathbf{R}_t = \begin{bmatrix} \cos(\psi_t) & -\sin(\psi_t) \\ \sin(\psi_t) & \cos(\psi_t) \end{bmatrix}. \tag{7}$$

In order to perform SLAM we need to establish a relationship between robot and world coordinates. We denote the location of a landmark i in the map corresponding with measurement j at time t as

$$\mathbf{m}_{i,t} = \begin{bmatrix} x_w \\ y_w \end{bmatrix} \quad \text{and} \quad \mathbf{z}_{j,t} = \begin{bmatrix} x_r \\ y_r \end{bmatrix}. \tag{8}$$

The measurement $\mathbf{z}_{j,t}$ will always be as the robot observes the landmark in robot coordinates, and the landmark's location $\mathbf{m}_{i,t}$ will always be given in world coordinates. The transformation between robot and world coordinates is given by the measurement equation

$$\mathbf{z}_{j,t} = \mathbf{h}(\mathbf{x}_t, \mathbf{m}_{i,t}) = \mathbf{R}_t^T \begin{bmatrix} x_w - x_t \\ y_w - y_t \end{bmatrix}, \tag{9}$$

or inversely,

$$\mathbf{m}_{i,t} = \mathbf{h}^{-1}(\mathbf{x}_t, \mathbf{z}_{j,t}) = \mathbf{R}_t \begin{bmatrix} x_r \\ y_r \end{bmatrix} + \begin{bmatrix} x_t \\ y_t \end{bmatrix}. \tag{10}$$

Exactly which measurement corresponds to which landmark in the map, as matched with the feature descriptors and confirmed with the outlier detection scheme, is stored in a correspondence vector $\mathbf{c}_t$.

III. Motion Model

Now that we have established a measurement equation, we need to derive a motion model for our robot so that we can perform SLAM. We use the velocity motion model. At every time step the controller of the robot will give it a forward and angular velocity,

$$\mathbf{u}_t = \begin{bmatrix} v \\ \dot{\psi} \end{bmatrix} + \mathcal{N}(\mathbf{0}, \mathbf{M}_t), \tag{11}$$

with v the forward translational speed and $\dot{\psi}$ the angular velocity. To characterize the uncertainty we add zero mean Gaussian noise with covariance matrix

$$\mathbf{M}_t = \begin{bmatrix} \alpha_1 v^2 + \alpha_2 \dot{\psi}^2 & 0 \\ 0 & \alpha_3 v^2 + \alpha_4 \dot{\psi}^2 \end{bmatrix}, \tag{12}$$

as is common practice [1]. The α parameters are robot and environment specific, and have to be estimated with practical testing and some degree of guesswork.

To update the robot states with the control input we define the motion equation as

$$\mathbf{x}_t = \mathbf{g}(\mathbf{x}_{t-1}, \mathbf{u}_t) = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \psi_{t-1} \end{bmatrix} + \begin{bmatrix} \mathbf{R}_{t-1} \begin{bmatrix} vT\cos(\dot{\psi}T) \\ vT\sin(\dot{\psi}T) \end{bmatrix} \\ \dot{\psi}T \end{bmatrix}, \tag{13}$$

with T the sample period of the system. Although this is an approximation, the accuracy lost due to the approximation is far smaller than the effect of expected noise in the control input $\mathbf{u}_t$.

IV. SLAM WITH THE RAO-BLACKWELLIZED PARTICLE FILTER

The particle filter can be used to approximate any distribution, and it is often utilized to accurately estimate non-Gaussian systems. A major drawback of the particle filter, however, is that with high dimensional problems a large number of particles is needed to describe the distribution sufficiently. The Rao-Blackwellized particle filter has been developed to overcome this problem [3]. This filter uses particles to describe some states and Gaussian distributions to represent all other states. In order to utilize it we need to factorize the SLAM problem as

$$p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_{i=1}^{n} p(\mathbf{m}_i \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}). \tag{14}$$

With this factorization we describe the required posterior as a product of $n+1$ probabilities. If we suppose that the exact location of the robot is known, it is reasonable to assume that the landmark positions are independent from one another and can therefore be estimated independently. Naturally, we do not know the robot's location, but this independence can be utilized when we use particles to estimate the robot position. It can even be shown that the above factorization is exact and not an approximation [4].

FastSLAM uses a particle filter to compute the posterior over robot states, $p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t}, \mathbf{u_{1:t}})$, and a separate EKF for every landmark in the map to obtain $p(\mathbf{m}_i \,|\, \mathbf{z}_{1:t}, \mathbf{u_{1:t}})$. What this means is that, instead of only one filter, we factor the problem into $1 + nm$ filters, where m is the number of particles. The large number of filters may seem excessive, but because of the low dimensionality of each individual filter the algorithm is remarkably efficient.

We define every particle to have a state vector for the robot states, and a mean vector and covariance matrix for every landmark, as

$$Y_t^{[k]} = \left\langle \mathbf{x}_t^{[k]}, \langle \mathbf{m}_{1,t}^{[k]}, \mathbf{\Sigma}_{1,t}^{[k]} \rangle, \ldots, \langle \mathbf{m}_{n,t}^{[k]}, \mathbf{\Sigma}_{n,t}^{[k]} \rangle \right\rangle, \tag{15}$$

with $\mathbf{x}_t^{[k]}$ the robot location and orientation for particle k, and $\langle \mathbf{m}_{i,t}^{[k]}, \mathbf{\Sigma}_{i,t}^{[k]} \rangle$ the i-th landmark's Gaussian mean and covariance. The FastSLAM algorithm, as it is executed at every time step, is given below in Algorithm 1. We proceed with a step by step explanation.

Algorithm 1 FastSLAM($Y_{t-1}, \mathbf{u}_t, \mathbf{z}_t, \mathbf{c}_t$)

1: **for** all particles $k \in \{1, 2, \ldots, m\}$ **do**
2: $\quad \mathbf{x}_t^{[k]} \sim p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}^{[k]}, \mathbf{u}_t)$
3: $\quad$ **for** all observed landmarks $\mathbf{z}_{i,t}$ **do**
4: $\quad\quad j = c_{i,t}$
5: $\quad\quad$ **if** landmark j has never been seen **then**
6: $\quad\quad\quad \mathbf{m}_{j,t}^{[k]} = \mathbf{h}^{-1}(\mathbf{x}_t^{[k]}, \mathbf{z}_{i,t})$
7: $\quad\quad\quad \mathbf{H}_j = \mathbf{J}_\mathbf{h}(\mathbf{m}_{j,t}^{[k]})$
8: $\quad\quad\quad \mathbf{\Sigma}_{j,t}^{[k]} = (\mathbf{H}_j^{-1})\mathbf{Q}_i(\mathbf{H}_j^{-1})^T$
9: $\quad\quad$ **else**
10: $\quad\quad\quad \hat{\mathbf{z}} = \mathbf{h}(\mathbf{x}_t^{[k]}, \mathbf{m}_{j,t}^{[k]})$
11: $\quad\quad\quad \mathbf{H}_j = \mathbf{J}_\mathbf{h}(\mathbf{m}_{j,t}^{[k]})$
12: $\quad\quad\quad \mathbf{Q} = \mathbf{H}\mathbf{\Sigma}_{j,t-1}^{[k]}\mathbf{H}^T + \mathbf{Q}_i$
13: $\quad\quad\quad \mathbf{K} = \mathbf{\Sigma}_{j,t-1}^{[k]}\mathbf{H}_j^T\mathbf{Q}^{-1}$
14: $\quad\quad\quad \mathbf{m}_{j,t}^{[k]} = \mathbf{m}_{j,t-1}^{[k]} + \mathbf{K}(\mathbf{z}_{i,t} - \hat{\mathbf{z}})$
15: $\quad\quad\quad \mathbf{\Sigma}_{j,t}^{[k]} = (\mathbf{I} - \mathbf{K}\mathbf{H_j})\mathbf{\Sigma}_{j,t-1}^{[k]}$
16: $\quad\quad\quad w^{[k]} = w^{[k]} f(\mathbf{Q}, \mathbf{z}_{i,t}, \hat{\mathbf{z}})$
17: $\quad\quad$ **end if**
18: $\quad$ **end for**
19: $\quad$ **for** all other landmarks $j' \notin \mathbf{c}_t$ **do**
20: $\quad\quad \mathbf{m}_{j',t}^{[k]} = \mathbf{m}_{j',t-1}^{[k]}$
21: $\quad\quad \mathbf{\Sigma}_{j',t}^{[k]} = \mathbf{\Sigma}_{j',t-1}^{[k]}$
22: $\quad$ **end for**
23: **end for**
24: **for** all $k \in \{1, 2, \ldots, m\}$ **do**
25: $\quad$ draw random particle k with probability $\propto w^{[k]}$
26: $\quad$ include $\left\langle \mathbf{x}_t^{[k]}, \langle \mathbf{m}_{1,t}^{[k]}, \mathbf{\Sigma}_{1,t}^{[k]} \rangle, \ldots, \langle \mathbf{m}_{n,t}^{[k]}, \mathbf{\Sigma}_{n,t}^{[k]} \rangle \right\rangle$ in Y_t
27: **end for**
28: **return** Y_t

- **Lines 1 and 2:** As with a normal particle filter, the FastSLAM algorithm begins by entering a loop over all the particles. The control input is used to sample a new robot pose for every particle according to the uncertainty in the motion model. We add random noise drawn from a zero mean Gaussian distribution with a covariance of $\mathbf{M}_t$, given in Equation 12, to the control input and use the motion equation $\mathbf{g}$, given in Equation 13, to find the new location and orientation of each particle.
- **Lines 3 and 4:** For every particle we enter a loop over all the measured landmarks. For every iteration the algorithm can do one of two things: add a new landmark, or update

an old landmark. The index of an old landmark in the map is given by the correspondence vector.

- **Lines 5 to 8:** A new landmark is added to the map using the measurement equation $\mathbf{h}$, given in Equation 9, to calculate its location in world coordinates. Since we want to use an EKF to estimate each landmark we have to linearize the measurement model by using a first order Taylor approximation with the Jacobian

$$\mathbf{J_h}(\mathbf{x}_t, \mathbf{m}_{j,t}) = \begin{bmatrix} \frac{\partial x_r}{\partial x_w} & \frac{\partial x_r}{\partial y_w} & \frac{\partial x_r}{\partial z_w} \\ \frac{\partial y_r}{\partial x_w} & \frac{\partial y_r}{\partial y_w} & \frac{\partial y_r}{\partial z_w} \\ \frac{\partial z_r}{\partial x_w} & \frac{\partial z_r}{\partial y_w} & \frac{\partial z_r}{\partial z_w} \end{bmatrix}. \quad (16)$$

  With this Jacobian we transform the uncertainty in measurement to an uncertainty in world coordinates.
- **Lines 9 to 15:** If a landmark has been observed before, we use the normal EKF equations to update its state vector and covariance. The state estimate is calculated by using the measurement model. The measurement model is then linearized with a Jacobian similar to the one used for new landmarks.
- **Line 16:** Once the landmark has been updated by using the measurement we have to calculate its effect on the weighting of the particle in question. As with a normal particle filter the importance weight is given by

$$w^{[k]} = \frac{\text{target distribution}}{\text{proposal distribution}}. \quad (17)$$

  The weighting function used in the algorithm can be shown [4] to be

$$f(\mathbf{Q}, \mathbf{z}_{i,t}, \hat{\mathbf{z}}) = |\mathbf{Q}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{z}_{i,t}-\hat{\mathbf{z}})^T \mathbf{Q}^{-1}(\mathbf{z}_{i,t}-\hat{\mathbf{z}})}. \quad (18)$$

  It is not necessary to update the weight for new landmarks as they will be the same for all particles, and therefore have no overall effect.
- **Lines 19 to 22:** If a previously observed feature has not been observed at the current time step its state vector and uncertainty will remain unchanged. All unobserved landmarks are therefore essentially ignored. This property of the algorithm is especially useful when a large map is maintained, as the number of unseen landmarks in the map does not impact the execution time.
- **Lines 24 to 27:** Resampling is done by drawing particles with a probability proportional to their normalized weights. Particles with low weights will be more likely to perish while particles with high weights will be copied and used at the next time step.
- **Line 28:** Finally the updated and resampled particles are returned to be used at the next time step.

A powerful possibility emerging from the use of particles is that of multiple hypothesis tracking. What it entails is that, since particles represent possible paths that the robot could have taken, we can calculate landmark correspondences for each particle separately. Because of the expensive nature of calculating feature matches we decide against this procedure and, instead, calculate one correspondence vector for all the particles. It is, however, important to note that the algorithm creates this possibility and future extensions can explore this feature.

V. Simulation

In order to test our SLAM systems we created a simulation environment that provides a realistic representation of the real world while facilitating a quantitative evaluation of the performance of the system.

A. Simulation environment

We created the environment with the aim of simulating the real world without it being unnecessarily complicated. We opted for a route through a corridor-like environment with landmarks on the walls. Although these landmarks are more structured than they typically would be in a real world situation, the structure should not influence the result significantly and should have the benefit of being easy to evaluate visually.

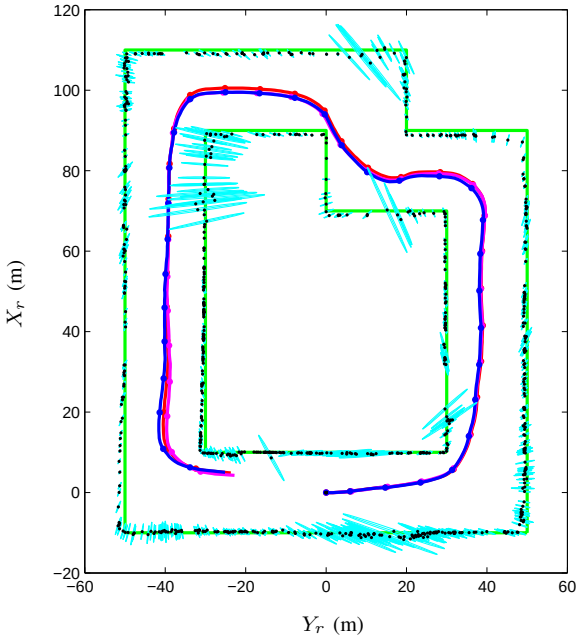
In order to create a control input we supply waypoints for the simulated robot to follow. At each time step a simple gain controller generates an input command that steers it towards the next waypoint. This control input is stored for use in the SLAM simulations but, before the robot executes the command, we add some Gaussian noise to simulate the uncertainty that we know exists in this process (in other words, we add process noise to the control input). The robot's actual motion from the noisy control is used as a ground truth trajectory and to generate the measurements.

As the robot moves through the environment, landmarks in the robot's field of view are included in the measurement at every time step. Because feature detectors will sometimes see a landmark at one time step and not at the next, even if it is in the field of view, we add a probability that a landmark will be seen. We project the landmarks onto the image planes of two cameras fixed on the robot and then add Gaussian noise to the pixel coordinates. Each landmark is assigned a unique scalar to be used as a descriptor. By changing or mixing these descriptors in a measurement we can simulate feature mismatches and investigate their effect on the accuracy of the SLAM system.

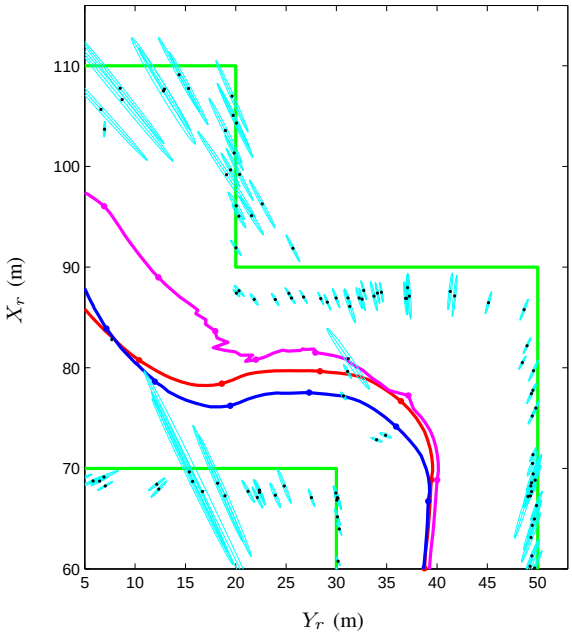
B. Simulation results

The simulation environment and the route and map as estimated by FastSLAM, using 250 particles, is depicted in Figure 2. At every time step each landmark has a 40% chance of being observed, but if it is observed, matching is done without error. When we display the route estimated by FastSLAM, we use a weighted average of the particles at every time step. In order to evaluate the accuracy we compare it to results obtained from another popular SLAM algorithm, namely EKF SLAM [8]. Results of the two algorithms are consistently similar in this simulation, even with varied noise parameters.

The experiment described above shows that it is possible to achieve accurate results using 250 particles with FastSLAM. To further investigate the relationship between the number



(a) simulation without landmark mismatches



(b) simulation with landmark mismatches

Fig. 2. The route and map from a simulation of FastSLAM, compared to EKF SLAM and ground truth (top). The bottom panel depicts an enlarged section of a simulation with landmark mismatches. The routes calculated with EKF SLAM are shown in magenta, the ground truth route in red and the environment walls in green. The estimated routes from FastSLAM are depicted in blue and the estimated landmark positions as black dots with corresponding confidence ellipses in cyan. Trajectories are shown with markers on every tenth time step.

of particles and accuracy we ran several simulations, each with a different number of particles. For every such number we ran the test 20 times in an attempt to remove the effect

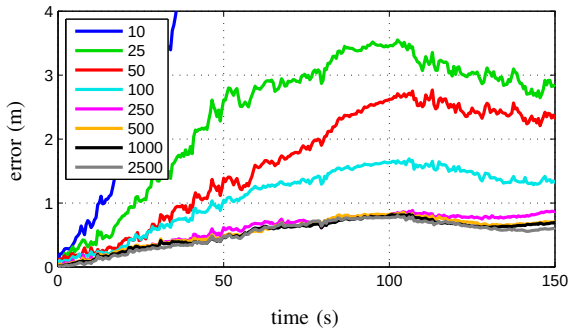


Fig. 3. The effect of different numbers of particles on the Euclidean error of the route estimated by FastSLAM.

of randomness introduced by the pose sampling step of the algorithm. Results of these experiments are shown in Figure 3.

We see that with FastSLAM in 2D, 250 particles is a good number to use as we do not lose much accuracy in comparison to using a larger number of particles.

In order to test the effect of landmark mismatches on the accuracy of FastSLAM we performed a simulation with such mismatches. The EKF SLAM algorithm is notorious for its inability to handle this kind of error [1] [8] and our simulation confirms this. With only six landmark mismatches over three time steps the EKF becomes unstable. With the same mismatches FastSLAM remains stable and introduces only a small degree of drift. This is a major practical advantage of the algorithm. These results are also depicted in Figure 2.

With these simulations we can establish, in a controlled environment, that FastSLAM achieves accuracy similar to EKF SLAM and is robust to landmarks mismatches. The following section describes our practical tests and results.

VI. Experimental results

The final step in our investigation and development of a FastSLAM system that uses stereo vision as a sensor is to test

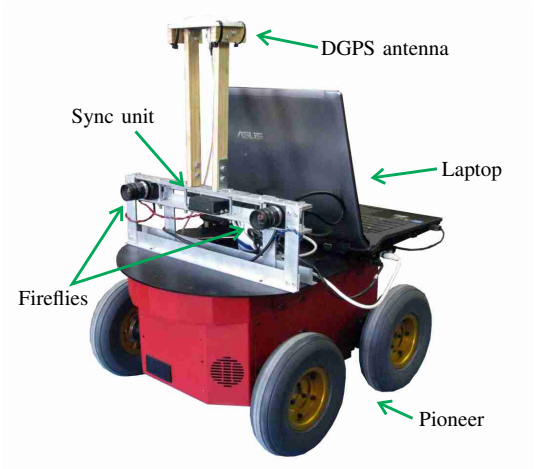


Fig. 4. Test platform.

Fig. 5. Sample frames (captured by the left camera) of the datasets used in our experiments.

the complete system with real world datasets.

A. Experimental setup and datasets

A real world dataset should ideally consist of a set of images captured by two synchronized and calibrated cameras, a control input and independently obtained ground truth location information that can be used to evaluate the performance of algorithms.

In order to capture such datasets we mounted a stereo camera set on a Pioneer 3-AT from Mobile Robots. We programmed the robot to execute a command given to it by a human using a joystick controller. At every time step we store the forward and rotational velocities so that they can be used as control input by the SLAM algorithms.

Our stereo camera rig consists of two Point Grey Firefly MV cameras with a synchronization unit we developed.

Ground truth data is recorded with a DGPS (accurate to about 5 cm) mounted on the robot. Note that this ground truth data is not used in our SLAM system, and is employed merely for evaluating results.

Figure 4 shows a picture of our test platform, indicating the various components.

When we work in a real world scenario we should expect problems such as bad lighting, uncluttered scenes (that give very few features), and a fair amount of shaking. We tried to capture realistic datasets that included these problems to a degree.

Two datasets were captured on the roof of the Electrical and Electronic Engineering building in Stellenbosch. The roof is a suitable environment to test 2D SLAM algorithms, since it is more or less flat. Apart from background trees moving in the wind it is also completely static.

The first of the two roof datasets includes a fair amount of maneuvering around two obstacles over a distance of about 45 metres. The second dataset comprises of a slow turn, a fairly long straight section, a three point turn with some reversing, and another straight section. The robot covered about 70 metres. Note that turning increases the process noise substantially because of wheel slippage.

A few frames of the datasets captured by one of the cameras are shown in Figure 5.

B. Experimental results

We show the results obtained from two experiments. The first was done using SURF features on the first dataset, and the second using SIFT features on the second dataset. These results are depicted in Figure 7 with corresponding location errors in Figure 6. We see that the Euclidean error from the first experiment grows over time. Drift is something that will be present with any localization system that does not employ absolute measurements (like GPS). In our work we attempt to limit this drift as much as possible.

We see that both SIFT and SURF can be used to obtain accurate results. Although we have no way of measuring the accuracies of the estimated maps, we can observe some structure and large quantities of landmarks located on the obstacles around which the robot moved.

VII. Conclusions

In this paper we investigated the use of the FastSLAM algorithm with landmarks originating from stereo image features. We explained how image features can be used as landmarks, with associated uncertainties in the form of Gaussian distributions. A measurement function converts features relative to the robot to landmarks in world coordinates and these landmarks are then matched over time, and outliers are identified and rejected. The FastSLAM algorithm then uses a particle filter to maintain the robot states, and for each particle a set of separate EKFs to estimate landmark locations.

We tested the system in a controlled simulation environment, and found that FastSLAM can be as accurate as EKF SLAM (when landmark matches are uncontaminated) but has the advantage of being largely unaffected by landmark

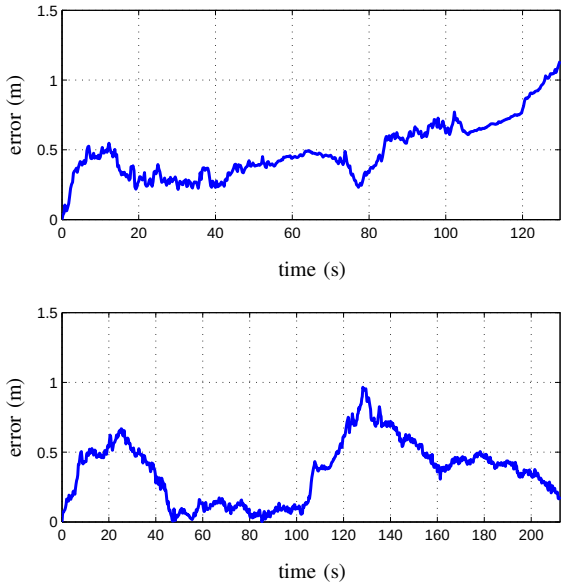


Fig. 6. The Euclidean error over time, as measured against DGPS, of the FastSLAM system using SURF features on the first dataset (top) and SIFT features on the second dataset (bottom).

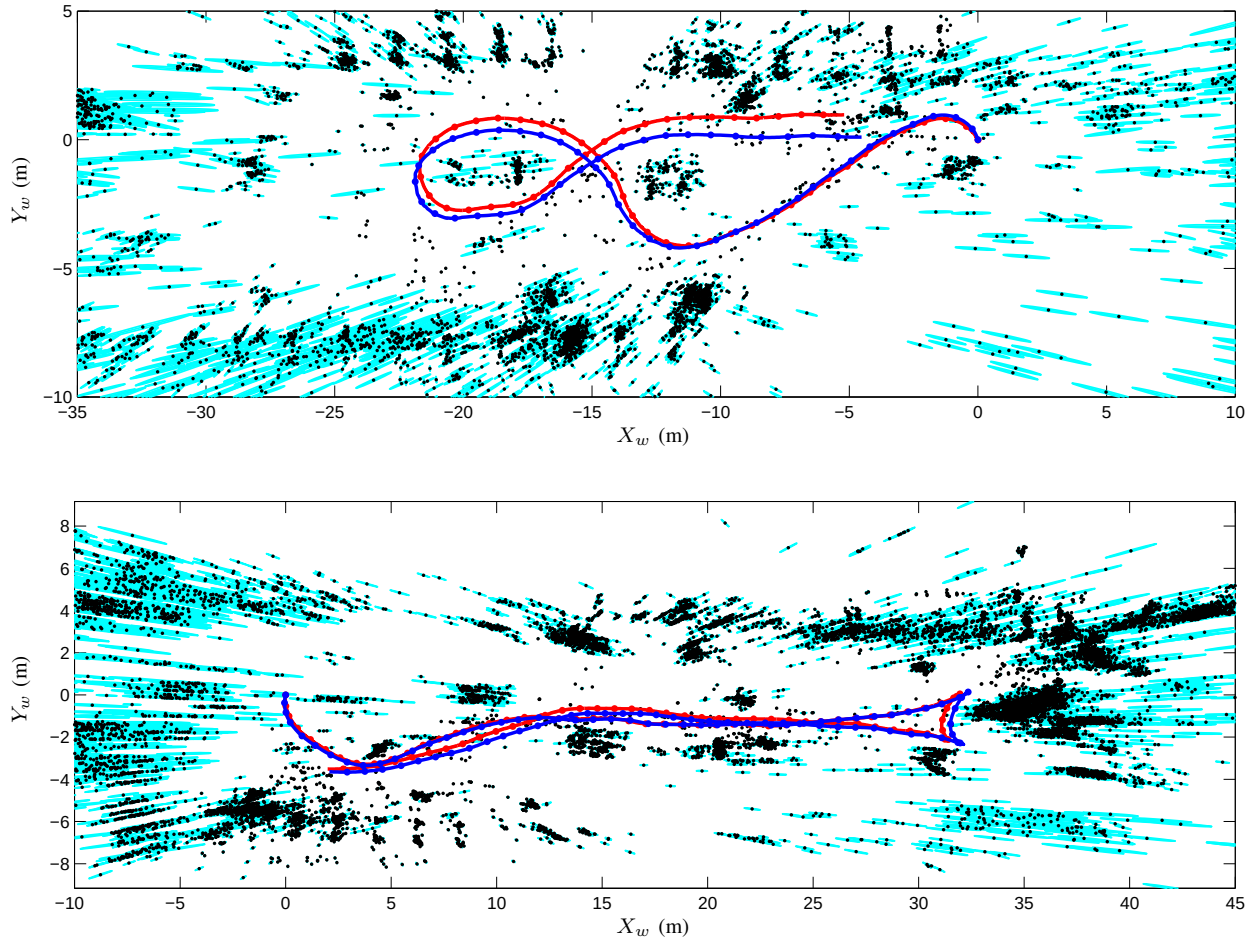


Fig. 7. Estimated routes (in blue starting at the origin) and maps from the FastSLAM algorithm using SURF features on the first outdoor roof datasets (top) and SIFT features on the second (bottom) with the DGPS ground truth in red. Markers are placed at every tenth time step of the routes. The landmarks that we show, as black dots with cyan confidence ellipses, are those that were observed on multiple time steps, i.e. those that contributed to the accuracy of the route estimation.

mismatches. This advantage of FastSLAM is significant, particularly when stereo features are used as landmarks, due to the unavoidable possibility of mismatches occurring. The problem of mismatches is inherent to image features, that often exhibit ambiguous characteristics, and we must therefore be able to rely on the SLAM system to remain stable in spite of such errors.

We also tested our complete FastSLAM system on data captured by a real robot. The accuracies achieved with either SIFT or SURF features are comparable to other state-of-the-art systems [10] [11].

We conclude that, because of its accuracy and robustness, FastSLAM can be a very effective algorithm to use with measurements from a stereo vision sensor.

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2006.

[2] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping (SLAM): Part I," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[3] A. Doucet, J. de Freitas, K. Murphy, and S. Russel, "Rao-Blackwellized particle filtering for dynamic Bayesian networks," *Conference on Uncertainty in Artificial Intelligence*, pp. 176–183, 2000.

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

[5] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[6] D. Lowe, "Object recognition from local scale invariant features," *IEEE International Conference on Computer Vision*, pp. 1150–1157, 1999.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[8] W. Brink, C. van Daalen, and W. Brink, "Stereo vision as a sensor for EKF SLAM," *22nd Annual Symposium of the Pattern Recognition Association of South Africa*, pp. 19–24, 2011.

[9] ——, "Probabilistic outlier removal for robust landmark identification in stereo vision based SLAM," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2822–2827, 2012.

[10] G. Dubbelman, W. van der Mark, and F. Groen, "Accurate and robust ego-motion estimation using expectation maximization," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3914–3920, 2008.

[11] J. Civera, O. Grasa, A. Davison, and J. Montiel, "1-point RANSAC for EKF-based structure from motion," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3498–3504, 2009.