FASTSLAM USING SURF FEATURES: AN EFFICIENT IMPLEMENTATION AND PRACTICAL EXPERIENCES

Peer Neubert, Niko Sünderhauf, Peter Protzel

Department of Electrical Engineering
and Information Technology
Chemnitz University of Technology
09111 Chemnitz
Germany
peer.neubert@informatik.tu-chemnitz.de
{*niko.suenderhauf, peter.protzel*}*@etit.tu-chemnitz.de*

Abstract: This paper describes how the recently published SURF features can be used as landmarks for an online FastSLAM algorithm that simultaneously estimates the robot pose and the pose of a large number of landmarks. An implementation with particular focus on efficient data structures like two-stage landmark data base and special balanced binary trees is described. Practical results on outdoor data sets at 3 Hz with about 3-6 % error of total traveled distance are shown.

1. INTRODUCTION

An online solution to the SLAM problem seems to be a key requirement for the practical (and commercial) use of robots in large natural environments. Besides classical approaches that use laser range finders or sonar sensors, vision based slam or visual odometry approaches emerged during the past years. A recent work that uses visual landmarks for SLAM is the work of (Barfoot, 2005). He uses SIFT features by (Lowe, 2004) as stable and recognizable landmarks but has to use a specialized FPGA to calculate the features and their descriptors online.
We seek for an efficient but powerful SLAM algorithm that is able to work online in outdoor environments for a longer period of time, without using specialized hardware. As the recently published SURF approach was found to be superior to SIFT in terms of runtime efficiency and keypoint quality by (Bay *et al.*, 2006), it should make an excellent landmark detector in the sought SLAM context. To enable the algorithm to run for longer periods of time and with a huge database of landmarks, special care has to be taken in the implementation. Before we present an efficient way to handle large amount of landmark data and prove the algorithm's performance with real-world tests, we shortly introduce the concepts of FastSLAM and SURF.

2. RELATED WORK

2.1 FastSLAM

FastSLAM (Thrun *et al.*, 2005) is a particle filter based approach to the simultaneous localization and mapping problem presented in (Montemerlo *et al.*, 2003). FastSLAM uses the conditional independence between any two disjoint sets of landmarks in a map, given the robot pose, to overcome exponential scaling. The original FastSLAM 1.0 algorithm samples over the robot pose and data association for a single observed feature. FastSLAM 2.0 improves the sampling step by taking the measurement into account while sampling a new pose. By using efficient data structures FastSLAM requires a map update time of $O(M \log N)$, where M is the number of particles and N is the number of the landmarks in the map. The authors of (Thrun *et al.*, 2005) presented some experiences using FastSLAM with occupancy grid maps. Some results of using FastSLAM for online visual motion estimation are presented in (Barfoot, 2005), where a stereo camera and visual landmarks are used. They were able to generate position estimates at 3Hz with an error of 4% of totaly traveled distance. They used a FPGA to extract SIFT features from pairs of stereo images, while computing the rest in software. Our solution achieves similar good results without special hardware assistance by using SURF features as visual landmarks.

2.2 SURF

(Bay *et al.*, 2006) presented SURF as a scale- and rotation-invariant interest point detector and descriptor. The SURF detector finds keypoints by using a so called Fast-Hessian Detector that bases on an approximation of the Hessian matrix for a given image point. The responses to Haar wavelets are used for orientation assignment, before the keypoint descriptor is formed from the wavelet responses in a certain surrounding of the keypoint. (Bauer *et al.*, 2007) showed that SURF outperforms SIFT in terms of speed and accuracy.

3. USING FASTSLAM WITH SURF

3.1 Adapted FastSLAM

We derive our adapted algorithm from FastSLAM 1.0 that was published by (Thrun *et al.*, 2004). The biggest change is the number of simultaneously observed features. The original algorithm works theoretically with a single observation per update step, while we increased this number for practical reasons (e.g. observing 100 SURF features at a time). The weighting step must be extended to cope with this larger number of landmarks. A large number of simultaneous observations makes it possible to reject some outliers. However, observing more potential landmarks at a time increases the costs for data association. Data association is therefore only done once per update step for the whole set of particles.

3.2 World, Observation and Motion Model

We work in a three dimensional cartesian world with the origin located at the robot's starting position. To determine the three dimensional coordinates of the observed landmarks, we extract the SURF features in both images taken by a stereo camera. Both sets of features are associated by using the descriptor and the constraints of epipolar geometry. The number of outliers produced during this step is marginal.

Because of coplanar image planes we can use the following triangulation matrix to get the three dimensional coordinates $\begin{pmatrix} x/s & y/s & z/s \end{pmatrix}^T$ from stereo coordinates $\begin{pmatrix} u & v & d & 1 \end{pmatrix}^T$ by using the homogeneous equation:

$$\begin{pmatrix} x \\ y \\ z \\ s \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -p_x \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/B & 0 \end{pmatrix} \begin{pmatrix} u \\ v \\ d \\ 1 \end{pmatrix} \quad (1)$$

Here u and v are the coordinates of the feature in the left image and d is the *disparity*, the difference of horizontal coordinates in the left and the right image. p_x, p_y, f and B are internal and external camera parameters.
To add landmarks to a database of already known landmarks we need to compute the coordinates in the world with respect to the robot pose $X_t = \begin{pmatrix} x & y & \phi \end{pmatrix}$. To calculate a landmark's world coordinates $\begin{pmatrix} x_{\mathrm{w}} & y_{\mathrm{w}} & z_{\mathrm{w}} \end{pmatrix}^T$ from local coordinates $\begin{pmatrix} x_{\mathrm{r}} & y_{\mathrm{r}} & z_{\mathrm{r}} & 1 \end{pmatrix}^T$ in the robot centered coordinate system, we can use a simple homogeneous transformation G:

$$\begin{pmatrix} x_{\mathrm{w}} \\ y_{\mathrm{w}} \\ z_{\mathrm{w}} \end{pmatrix} = \begin{pmatrix} -\sin\phi & 0 & -\cos\phi & x \\ \cos\phi & 0 & -\sin\phi & y \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_{\mathrm{r}} \\ y_{\mathrm{r}} \\ z_{\mathrm{r}} \\ 1 \end{pmatrix} \quad (2)$$

Because the robot pose may be different for different particles, every particle contains its own database of landmarks. A landmark is represented by a mean μ and a covariance matrix Σ. It is handled by an Extended Kalman filter.
Hence, a single particle is of the form

$$\left[\quad \begin{pmatrix} x & y & \phi \end{pmatrix}^T_{1:\mathrm{t}} \qquad (\mu, \Sigma)_{1:\mathrm{N}} \quad \right]$$

In every update step the robot pose $X_{\mathrm{t}} = \begin{pmatrix} x & y & \phi \end{pmatrix}$ is updated for every particle. First we estimate the current pose from the odometric sensor's data $\begin{pmatrix} \Delta x_{\mathrm{O}} & \Delta y_{\mathrm{O}} & \Delta \phi_{\mathrm{O}} \end{pmatrix}$. The overall estimated traversed distance thus is
$d = \sqrt{\Delta x_{\mathrm{O}}^2 + \Delta y_{\mathrm{O}}^2}$
So we can sample new positions for each particle by using this data and a random factor following a normal distribution that reflects the errors in the odometric sensors:

$$x_{\mathrm{t}} = x_{\mathrm{t}-1} + d \cdot \cos(\phi_{\mathrm{t}-1} + \frac{1}{2}\Delta\phi_{\mathrm{O}}) \cdot (\sim \mathcal{N}\left(1, \sigma_{\mathrm{x}}^2\right))$$

$$y_{\mathrm{t}} = y_{\mathrm{t}-1} + d \cdot \sin(\phi_{\mathrm{t}-1} + \frac{1}{2}\Delta\phi_{\mathrm{O}}) \cdot (\sim \mathcal{N}\left(1, \sigma_{\mathrm{y}}^2\right))$$

$$\phi_{\mathrm{t}} = \phi_{\mathrm{t}-1} + \Delta\phi_{\mathrm{O}} \cdot (\sim \mathcal{N}\left(1, \sigma_{\phi}^2\right))$$

$\sigma_{\mathrm{x}}^2, \sigma_{\mathrm{y}}^2$ and σ_{ϕ}^2 are the variances of the according dimension of odometric data.

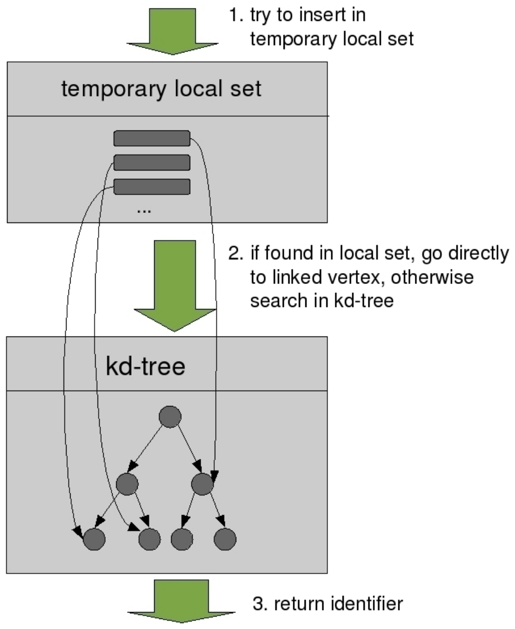
1. try to insert in temporary local set
temporary local set
...
2. if found in local set, go directly to linked vertex, otherwise search in kd-tree
kd-tree
3. return identifier

Fig. 2. A schema of two-stage data association. A currently observed feature is searched in the temporary local set. If it is found, we link directly to the corresponding vertex in the kd-tree otherwise we search the feature in the kd-tree.

As common in particle filters, these particles are resampled after getting weighted in a second step. The weighting occurs through consistence with landmark observations.

3.3 Using SURF Descriptors for Data Association

The original FastSLAM algorithm observes one single landmark per update step and samples over robot pose and data association. Our modification observes a larger number of landmarks simultaneously and makes data association once for all particles. The data association between currently observed features and landmarks from the data base is done with the SURF descriptor. To speed up the data association step, we associate the currently observed and already known landmarks in a two-step scheme that is illustrated in fig. 2.
The landmark descriptors are not saved in the particles but are used to find landmark identifiers in the data structures shown in fig. 2. These identifiers are easy to compute and are used to find the (μ, Σ) pose representation of the landmarks in the particles. The landmark poses are saved in a tree structure in the particles. We will have a closer look at this tree structure later in this section. The identifiers are *not* simple links into that tree but rather encode the descent that is necessary to find the corresponding landmark pose.
To speed up the data association, the database of landmark descriptors is split in two parts: A temporary local set that serves as a kind of cache for landmarks that have been observed during the last cylce, and a kd-tree structure for older landmarks. The descriptors of the currently observed landmarks are first searched in the temporary local set. If no match is found, the search continues in the kd-tree. This temporary cache significantly speeds up the association step because fewer possible matches have to be checked. Furthermore, it reduces the number of false matches because landmarks observed in the last update step get a bigger chance of being associated in the current step. A possible drawback however is that the loop closing performance may decrease because the association between a currently observed landmark and an older, already known landmark is penalized. To avoid this, a very close correspondence (i.e. a small euclidian distance between the descriptor vectors) between the observed landmark and the candidate landmark from the temporary local set is required before they get associated. In particular, the upper limit for an association in the temporary local set is smaller than in the kd-tree.
Whenever a new landmark has been observed (i.e. it was neither found in the local set nor in the kd-tree), a new vertex is created in the kd-tree. This vertex contains the SURF descriptor (which also serves as key during the search in the tree) and the newly created unique identifier. An entry for the new landmark is created in the local set as well. The entry in the local set is linked to the new vertex in the kd-tree as illustrated in fig. 2.
If a landmark is found in the temporary local set or directly in the kd-tree, the corresponding identifier is saved for the following particle update step.
Outliers in the data association (i.e. false matches) are detected on a per-particle basis. A match is discarded if the difference of the estimated position and the current measurement is too large.

3.4 The Particles

Each particle contains a balanced binary search tree for managing the landmark postion estimates. The tree is balanced by construction and need not to be reordered. Every vertex includes a position estimate for a single landmark. Landmarks are found by their position in the tree, the necessary descent operations can be decoded by very simple and fast bit operations from the identifier described above.

3.5 Resampling

To resample the particle set it is necessary to weight the particles according to how well the positions of the observed landmarks and their associated counterparts from the database correspond.

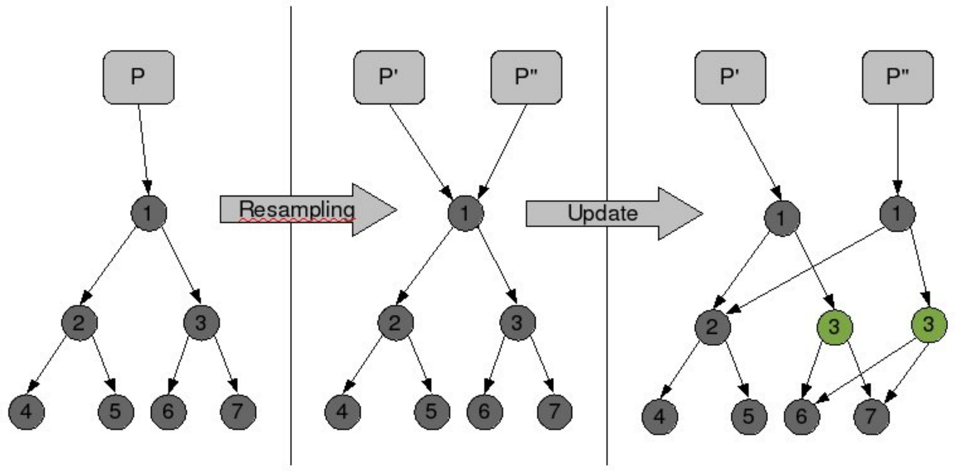

Fig. 1. Taking two samples P′ and P′′ from an origin particle P and afterwarts modifying a vertex in the binary tree in P′′. Therefor a new path is generated in P′′ and the rest of the origin tree is appended.

Therefore we weight every association and sum over these weights. (Thrun *et al.*, 2005) suggests to calculate the weight for a single observation by

$$w = \frac{1}{\sqrt{\det(2\pi Q)}} e^{-\frac{1}{2}(x-\mu)^T Q^{-1}(x-\mu)} \tag{3}$$

where μ is the mean of the landmark position estimate from the data base in the current robot coordinate system, x is the three dimensional coordinate vector of the observed feature in current robot coordinate system.
Q is calculated as follows:

$$Q = H\Sigma H^T + R \tag{4}$$

where H is the derivative of the inverted transformation matrix G with respect to the feature coordinates.
R is the covariance of the measurement, in our case

$$R = JSJ^T \tag{5}$$

Here S is the covariance caused by the discreetness of the pixels. We assumed:

$$S = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The Jacobian of the current stereo coordinate calculation matrix in our observation model is given by J:

$$J = \begin{pmatrix} -\frac{B}{d} & 0 & \frac{B \cdot (u - px)}{d^2} \\ 0 & -\frac{B}{d} & \frac{B \cdot (v - py)}{d^2} \\ 0 & 0 & \frac{f \cdot B}{d^2} \end{pmatrix} \tag{6}$$

with $\begin{pmatrix} u & v & d \end{pmatrix}^T$ as stereo coordinates of the current observation.
Through the resampling step, some new particles may have been resampled from the same original particle. To increase speed and avoid too many copy operations, these new particles share the same old binary search tree which contains the landmark positions. That is uncomplicated as long as the landmark positions in the tree are not changed by the particles. As this will definitively occur during the update step, the path to the changed vertex has to be regenerated for the respective particle. The rest of the tree can simply be appended. Because the tree is balanced, the resulting update costs for a single landmark are $O(\log N)$. Figure 1 illustrates how two particles P′ and P′′ share the same tree after resampling. During the update step, P′′ changes the position estimate of the landmark that is contained in vertex 3. Therefore, the path from the tree's root to vertex 3 has to be regenereated for P′′. Notice that the rest of the tree remains unchanged and is still shared between P′ and P′′.

4. PRACTICAL ASPECTS

4.1 Hardware

We tested our implementation on a mobile robot platform, based on a Pioneer 2AT equipped with a stereo camera. The computations were done on an Athlon64 3200+ machine.

4.2 Practical Constraints

The 3D information that can be derived from a stereo camera are prone to errors. Figures 3 and 4 show how the standard deviations for the estimated X and Z coordinate for a triangulated world point rise very fast with increasing distance from the camera. In general we can say that the further away a point is from the camera, the larger

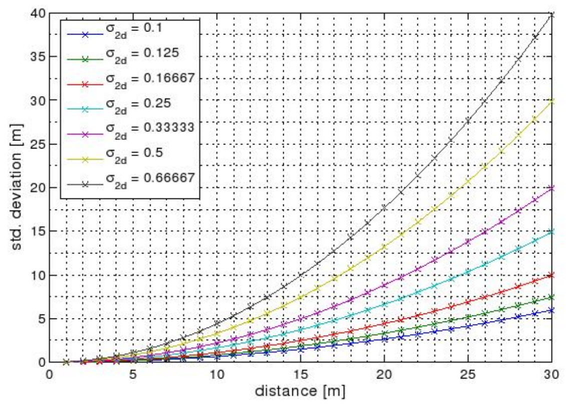

Fig. 3. Std. deviation for Z-coordinate (camera viewing direction) of an observed feature increases fast with rising distance

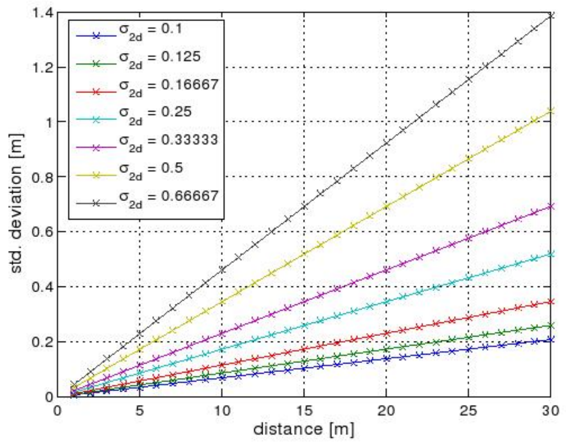

Fig. 4. Std. deviation for X-coordinate (perpendicular to camera viewing direction) of an observed feature increases slower than Z-coordinate with rising distance

the uncertainties are. Hence, we set a maximum range limit of about 24m by disregarding observations with dispartiy < 1.
As can be seen from the figures, the uncertainty in the direction the camera is facing increases much faster than in the directions perpendicular to it. If the camera is facing forward, i.e. in the direction the robot is moving, we are moving in the direction of largest uncertainty, which negatively affects the accuracy of every visual SLAM or visual odometry algorithm. The stereo camera should therefore not be facing into driving direction, but perpendicular to it. We expect a faster convergence of landmark position estimation.
Handling unusable pictures, that may occur while aperture adjustment is performed by the camera as shown in figure 5, is another problem at hand. Unusable images should be detected and handled carefully, for example by retaining the old local temporary set and skip the resampling step.

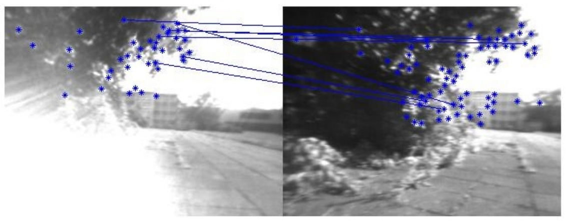

Fig. 5. Missmatches occur in unusable pictures, features attended for association are marked.

4.3 Practical Experiments and Results

We are able to run the algorithm online, with about 3 Hz. The effective runtime depends on the number of observed features, in our experiments we worked with about 120 observations per update step. Another influencing value is the total number of landmarks. Here, our two stage data association handled a database of up to 16.000 landmarks.
For first outdoor experiments we traversed a ca. 30 m long cyclic course on hard ground with a set of 100 particles. The standard deviations for the odometry data was estimated to be 10% for straight forward motion and 30% for rotation. These estimates are by far worse than the real odometry errors, but were set to higher values to prove our algorithm's performance. Odometry was spoiled with noise by adding systematic errors to motion estimation for the same reasons (e.g. 15% of $\Delta\phi_O$). Figure 6 shows the robot's position estimtated by our algorithm, spoiled odometry data, and ground truth. Ground truth was estimated from a sensor fusion between real odometry and a magnetic compass.
The results of several parameter settings with 100 particles decreased the position error from 20-40% (spoiled odometry) to 3-6% of the traveled distance. The second test dataset stems from a 120 m long outdoor traverse. Again, we used 100 particles and added simulated errors to the odometric sensors to better prove the algorithm's capabilities.
The resulting errors are comparable to the errors of the first testcase. Figure 7 shows a typical result for this test: The original way was a closed route (nearly like the estimation of our algorithm) the odometric data seems to be far off and the result of our algorithm improves position estimation significantly. A limitation is still the capability for loop closing. It's effected by a too small variance in the particle set at the time of loop closing.

5. FUTURE WORK

In future work we will adapt the improvements in position estimation as proposed by FastSLAM 2.0 and fuse odometry data with DGPS information or visual odometry before using it in the SLAM

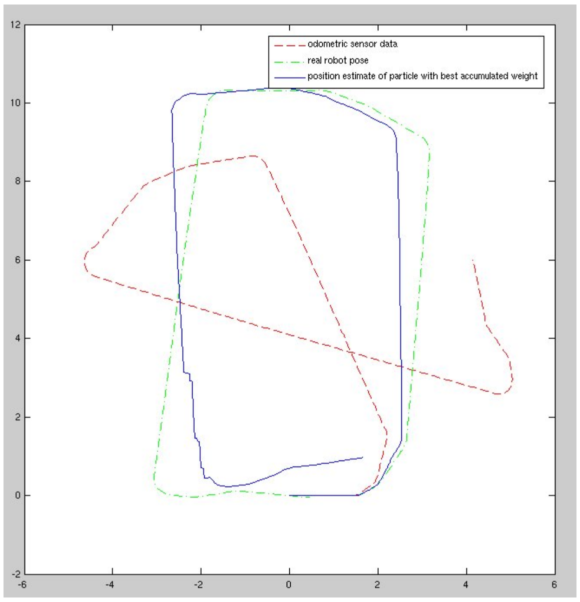

Fig. 6. Improvement of odometric sensor data with our algorithm on a short 30m traverse.

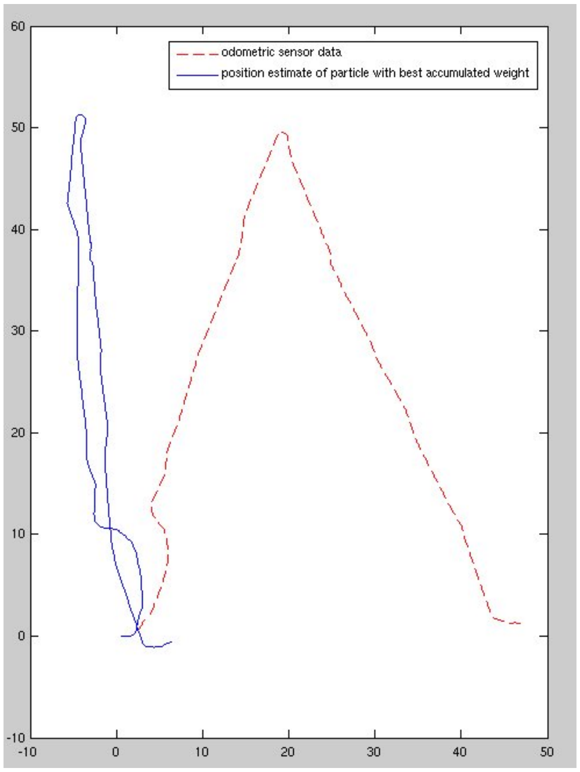

Fig. 7. Comparison of results on an about 120 m traverse of odometric alone and the particle with best accumulated weight.

algorithm.
The motion modell will be extended to six degrees of freedom to acquire a full 3D SLAM approach.
The efficiency of the data association step can be improved further by doing a position based association. Candidate landmarks that could be possible matches for a currently observed landmark can be pre-determined by regarding the current position estimate and the field of view of the camera. Ordering the landmarks in an octtree, and a simple visibility check of an landmark can help reducing the set of candidate landmarks.
The use of hash-functions to reduce the size of the global kd-tree (fig. 2) by splitting the set of landmarks into hash-classes seems to be a promising approach to accelerate data association when large databases of features are available.

6. CONCLUSION

We have demonstrated and described several aspects of using SURF features for SLAM. We described certain aspects of our fast implementation that uses efficient data structures. Our SURF-SLAM implementation runs completely in software with framerates of 3Hz. No special hardware like FPGAs are required. Practical experiences proved the algorithm's performance and capabilities.

REFERENCES

Barfoot, T.D. (2005). Online visual motion estimation using fastslam with sift features. In: *Proceedings of the International Conference on Robotics and Intelligent Systems (IROS).* Edmonton, Alberta.

Bauer, Johannes, Niko Sünderhauf and Peter Protzel (2007). Comparing several implementations of two recently published feature detectors. In: *Proceedings of the International Conference on Intelligent and Autonomous Vehicles, IAV07.* Tolouse, France.

Bay, Herbert, Tinne Tuytelaars and Luc Van Gool (2006). Surf: Speeded up robust features. In: *Proceedings of the ninth European Conference on Computer Vision.*

Lowe, David G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. In: *International Journal of Computer Vision, 60, 2.* pp. 91–110.

Montemerlo, M., S. Thrun, D. Koller and B. Wegbreit (2003). Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI).* Acapulco, Mexico.

Thrun, Burgard and Fox (2005). *Probabilistic Robotics.* The MIT Press. Cambridge, Massachusetts, London, England.

Thrun, S., M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto and E. Nebot (2004). Fastslam: An efficient solution to the simultaneous localization and mapping problem with unknown data association. *Journal of Machine Learning Research.*