LOCALIZATION OF MOBILE ROBOTS WITH OMNIDIRECTIONAL VISION USING PARTICLE FILTER AND ITERATIVE SIFT

Hashem Tamimi[1], *Henrik Andreasson*[2], *André Treptow*[1], *Tom Duckett*[2] *and Andreas Zell*[1]

{tamimi, treptow, zell}@informatik.uni-tuebingen.de, {henrik.andreasson, tom.duckett}@tech.oru.se
[1] Department of Computer Science, University of Tübingen, Tübingen, Germany
[2]AASS, Department of Technology, University of Örebro, Örebro, Sweden

ABSTRACT

The Scale Invariant Feature Transform, SIFT, has been successfully applied to robot localization. Still, the number of features extracted with this approach is immense, especially when dealing with omnidirectional vision. In this work, we propose a new approach that reduces the number of features generated by SIFT as well as their extraction and matching time. With the help of a particle filter, we demonstrate that we can still localize the mobile robot accurately with a lower number of features.

1. INTRODUCTION

Vision based robot localization demands image features with many properties. On one hand the features should exhibit invariance to scale and rotation as well as robustness against noise and changes in illumination. On the other hand they should be extracted very quickly so as not to hinder the other tasks that the robot plans to perform. Local descriptors are commonly employed in robot localization because they can be computed efficiently, are resistant to partial occlusion, and are relatively insensitive to changes in viewpoint.

SIFT features, explained in section 2, have been widely used in the robot localization field. In [1] the SIFT scale and orientation constraints are employed for matching stereo images; after matching the features, the authors used a least-squares procedure to reach better localization performance. In [2] a modified version of the SIFT approach is proposed and used to solve the robot localization problem; their approach takes the properties of panoramic images into consideration. The work in [3] proposes an approach to modeling the pose-dependent characteristics of the SIFT features; their model is a learning based one and is successfully applied to the robot localization problem. In order to further minimize the classification errors during localization, the work in [4] has proposed extracting SIFT features from each image and then using spatial relationships among the locations by means of a hidden Markov model. In [5] an image map based on SIFT and Harris corners is built and used later for localization.

In this paper we apply a new approach, iterative SIFT, to the robot localization problem. We try to reduce the computational effort of the feature extraction and matching process as much as possible while maintaining high localization accuracy. This means that the robot will try to localize itself using less features than with classical SIFT.

The remaining sections of this paper are organized as follows: Section 2 is a review of the SIFT approach. Section 3 introduces the idea behind iterative SIFT and presents its algorithm. In section 4 the computation time of SIFT is compared with that of iterative SIFT. Section 5 reviews the Monte Carlo approach to localization. Section 6 gives the experimental results of applying the iterative SIFT algorithm to the robot localization problem. Finally we conclude this paper in section 7.

2. SCALE INVARIANT FEATURE TRANSFORM

The Scale Invariant Feature Transform (SIFT), developed by Lowe [6], is invariant to image translation, scaling and rotation. SIFT features are also partially invariant to illumination changes and affine 3D projection. These features have been widely used in the robot localization field as well as many other computer vision fields.

The SIFT algorithm has 4 major stages:

1. **Scale-space extrema detection:** The first stage searches over scale space using a Difference of Gaussian (DoG) function to identify potential interest points.
2. **Keypoint localization:** The location and scale of each candidate point are determined and keypoints are selected based on measures of stability.
3. **Orientation assignment:** One or more orientations are assigned to each keypoint based on local image gradients.
4. **Keypoint descriptor:** A descriptor is generated for each keypoint from local image gradients information at the scale found in stage 2.

The first stage is clarified as follows: For each octave in the scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images. Adjacent Gaussian images are subtracted to produce the DoG images. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process is repeated. For a more detailed discussion of the keypoint generation and factors involved see [6].

SIFT features are distinctive and invariant features used to robustly describe and match digital image content between different views of a scene. While invariant to scale and rotation, and robust to other image transforms, the SIFT feature description of an image is typically large and slow to compute. For example, the work

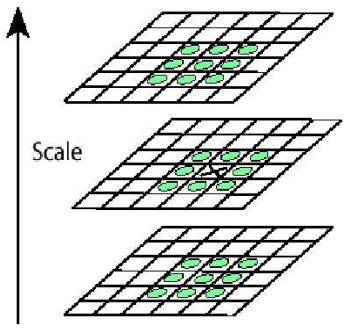

Fig. 1. Local extrema detection. The pixel marked $\times$ is compared against its 26 neighbours in a $3 \times 3 \times 3$ neighbourhood that spans adjacent DoG images (from [6]).

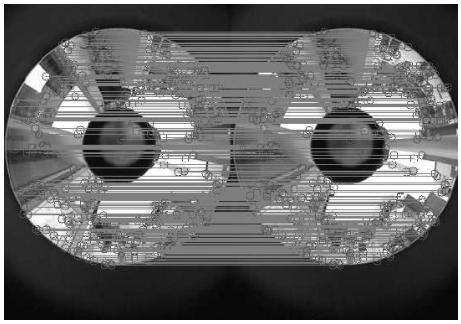

Fig. 2. Matching two different panoramic images using SIFT. The number of common features is huge, requiring a high computation time.

in [7] presents a study of SIFT features for outdoor robot localization. Although their approach is able to pick up features that are stable despite the varying illumination, the authors reported some disadvantages of using SIFT, specifically that it takes a long time to extract the features from an image. Furthermore, the number of features is immense, which poses problems when searching for the matching pairs, along with having to store a large amount of data. An example of this problem is illustrated in figure (2), where the robot approaches a known area and a large number of keypoints is matched between current and stored features. This problem arises because panoramic images naturally hold a rather high amount of information from all the surroundings, which makes the process of dealing with this amount of information very time-consuming.

3. ITERATIVE SIFT

The main objective of the iterative SIFT approach is to reduce the number of keypoints and their corresponding extraction and matching time, while maintaining the same descriptor for each keypoint. In the classical SIFT approach, keypoints are detected by testing each value in the DoG at each scale with the 8 surrounding values of the same scale as well as with 9 neighbouring values in the scale above and 9 neighbouring values in the scale below. The first and last DoG scales are not examined. This means $26 \times m \times n$ comparisons for a DoG of size $m \times n$, taking into consideration that points around a given border of each DoG are not included in the keypoint detection, as seen in figure (1). Since SIFT establishes multiple scales in each octave, the above analysis is applied several times to each scale in each octave. Each octave has one quarter of the pixels of the previous one, so that keypoint detection in lower octaves requires more time than in higher ones. We aim to modify this exhaustive search into a sample based one.

In the proposed approach, the number of keypoints can be defined in advance. The process of finding the keypoints continues iteratively without the need for sequentially going through the whole scale space. This involves two phases. The first phase is randomly searching the scale space for local extrema. The random search is followed by an update phase only when the local extremum is more likely to be found. The theory behind the iterative SIFT approach is mainly based on the assumption that local extrema points are located in a blob region [8], i.e. smooth wide two dimensional hills or valleys. In other words, blobs are regions in the image that are either significantly brighter or significantly darker than their surroundings. A local extremum cannot be located on a flat region and can hardly be found near it. Another possible location of local extrema are spikes, i.e. rapidly changing narrow regions. But since the scale space structure involves multiple smoothing operations on the image, only information on the coarse scale remains and the spikes are filtered out.

With the above assumption we can say that our search mechanism involves dealing only with two cases when searching for a local extremum: 1) In the case where we detect a blob region, an update phase handles the search for the position of the local extremum in that region. The search ends either when the local extremum is found or when a given number of trials elapses. 2) In the case where we detect a non-blob region, the result of the search in this area is ignored and the search is started somewhere else.

The test whether a point p lies in a blob region or not is shown as a function $isBlob()$ in equation (1), where T_{Scale} is a threshold depending on the scale of the point p.

$$isBlob = \begin{cases} True & (Abs(p) > T_{Scale}) \\ False & otherwise \end{cases} \quad (1)$$

In the iterative SIFT algorithm, see algorithm (1), the following functions are used: The function $KeypointDescriptor()$ builds the keypoint descriptor for the point located in (x_i, y_i). The function $RandomCoord()$ returns a set of randomly chosen (x,y) coordinates bounded by the size of the current scale. The function $isExtremum()$ tests whether the point located in (x_i, y_i) is a local maximum or local minimum. This involves comparing the value of a point with the 26 neighbours as explained above. The function $findNewCandidate()$ searches the 8 neighbours of the point (x_i, y_i) in the same scale and returns the coordinate of the maximum point if $p(x_i, y_i)$ is positive and the coordinate of the minimum point if $p(x_i, y_i)$ is negative. Note that the DoG image contains both positive and negative points with the median 0.

The algorithm is clarified as follows: We first initialize a set of samples with random numbers, each of which holds a value that represents the coordinate of one of the points in the current scale. The samples are then verified so that only those that have a value above the given threshold T_{Scale} remain. This reflects our assumptions that a value above this threshold is most probably a point that lies in a blob. The total number of samples in the algorithm after

Algorithm 1 The iterative SIFT algorithm.

Definitions:
$NSamples$: The number of samples.
$NKeys$: The number of requested keypoints.
$NTrials$: The number of trials.
$NScales$: The number of Scale images.
initialization: $Keypoints \leftarrow \{\}$;
for $Scale \leftarrow 1$ to $NScales$ **do**
$\{(x_i, y_i)\} \leftarrow$ RandomCoord(), given that
$isBlob\,(p\,(x_i, y_i))$ is True, $\forall i = 1, 2, ..., Nsamples$;
$i \leftarrow 0$;
while $i < NSamples$ And $|Keypoints| < NKeys$ **do**
$i \leftarrow i + 1$;
$trial \leftarrow 0$;
$ExtremaFound \leftarrow False$;
while $trial < NTrials$ And $\neg ExtremaFound$ **do**
if $isExtremum\,(x_i, y_i)$ **then**
$Key_i \leftarrow KeypointDescriptor(x_i, y_i)$;
$Keypoints \leftarrow Keypoints \cup \{Key_i\}$;
$ExtremaFound \leftarrow True$;
else
$(x_i, y_i) \leftarrow findNewCandidate\,(x_i, y_i)$;
end if
$trial \leftarrow trial + 1$;
end while
end while
end for

the verification should be $NSamples$ and depends on the size of the scale image. After some initial experiments, we found that setting $NSamples = r \times M \times N$ with $r = 3$ leads to best results. Here (M, N) are the size of the current scale image. The search for the keypoints in the algorithm is applied to one scale after another. The search stops when the required number of keypoints $NKeys$ is reached. The search within a blob involves testing whether the current point is a local extremum using the function $isExtremum()$. If the point is an extremum then the keypoint descriptor is generated for that point. If the current keypoint is not a local extremum, the function $findNewCandidate()$ returns a new coordinate for a candidate local extremum. With this update, the sample is assumed to move in the direction of the local extrema. The update goes on until the local extremum is found or until the maximum number of trials, $NTrials$, is reached. In the later case the point is neglected and the search is performed by another sample in another new location.

4. TIME CONSIDERATIONS OF ITERATIVE SIFT

Before introducing the experimental results with the particle filter, we first demonstrate the ability of iterative SIFT to extract a given number of keypoints and how this varies with the required time.

Figure (3) shows the feature extraction time when applying SIFT and iterative SIFT approaches to a sample image. The plot shows that the SIFT approach finds a constant number of keypoints and requires a relatively high and constant time.

It is worth noting that even though iterative SIFT finds a good amount of features in less time than it takes SIFT, our approach

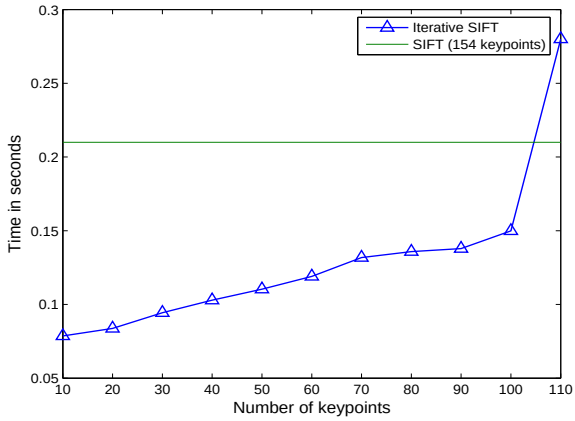

Fig. 3. The time required to extract a given number of keypoints on a sample images, including the time of calculating the DoG, done with a 3GHz Pentium.

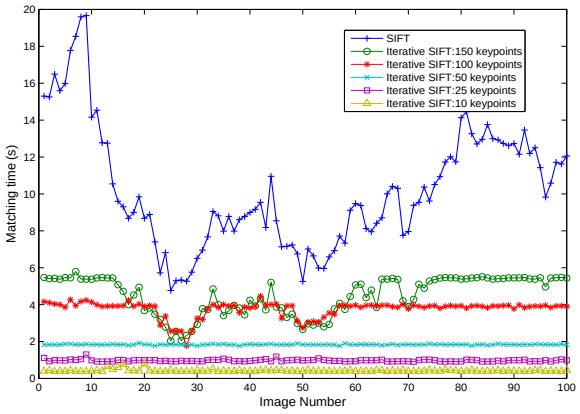

Fig. 4. The time required to extract and match a given number of keypoints using iterative SIFT versus the time of SIFT for a set of 100 images, done with a 3GHz Pentium.

fails to keep doing so as the number of given keypoints gets higher and is not able to find the number of keypoints that the classical SIFT finds within less or equal time. This is because our approach is based on samples which are randomly distributed in the search space. These samples need much more time than the linear approach of SIFT in order to cover the whole search space. Still, the application of robot localization, as well as other similar applications, does not need such a large number of keypoints as SIFT produces.

Since the number of keypoints using iterative SIFT is clearly reduced, the time for matching two sets of keypoints during the robot localization process can be minimized to a great extent. Figure (4) shows the feature extraction and matching time required by each approach. This time is constant in most cases unless the required number of keypoints cannot be found. This happens in images that contain small details.

5. MONTE CARLO LOCALIZATION

Monte Carlo methods or Particle Filters [9] have become quite popular in recent years for estimating the state of a system at a certain time based on the current and past measurements. The probability $p(X_t|Z_t)$ of a system being in the state X_t given a history of measurements $Z_t = \{z_0, ..., z_t\}$ is approximated by a set of N weighted particles:

$$S_t = \{x_t^{(i)}, \pi_t^{(i)}\},\ i = 1...N. \tag{2}$$

Each particle $x_t^{(i)}$ describes a possible state together with a weight $\pi_t^{(i)}$, which is proportional to the likelihood that the system is in this state. Particle Filtering consists of three main steps:

1. Create a new particle set S_{t+1} by resampling from the old particle set S_t based on the particle weights $\pi_t^{(i)}, i = 1...N$
2. Predict the next particle states based on the dynamic model $p(x_{t+1}^{(i)}|x_t^{(i)}, u_t)$ with odometry u_t, $i = 1...N$
3. Calculate the new weights by application of the measurement model: $\pi_{t+1}^{(i)} \propto p(z_{t+1}|X_{t+1} = x_{t+1}^{(i)}), i = 1...N$.

The estimate of the system state at time t is the weighted mean over all particle states:

$$\hat{X}_t = E(S_t) = \sum_{i=1}^{N} \pi_t^{(i)} x_t^{(i)}. \tag{3}$$

The weights $\pi_t^{(i)}$ are normalized so that $\sum_{i=1}^{N} \pi_t^{(i)} = 1$. In our case the state is described by a three dimensional vector $x_t = (x, y, \theta)_t$ containing the position of the robot (x, y) and the orientation θ. The x and y coordinates are initialized randomly within a radius of one meter around a randomly selected database position. To estimate an initial orientation θ, a 32 bin orientation difference histogram is created where the orientation difference is calculated from the matched points. The highest bin is then used to calculate the orientation with an added random value drawn from a normal distribution with standard deviation $\pi/8$ radians. The prediction and measurement steps are described in the following sections. In the experiments, a total of 500 particles were used and 10% of these were randomly reinitialized at each iteration to enable relocalization.

5.1. Dynamic Model

All state variables $x_t^{(i)} = (x, y, \theta)_t$ are updated from the odometry readings u_t from the robot. To cope with the additional uncertainty due to odometry error, the odometry values are updated with small random values drawn from a normal distribution, using a standard deviation of 0.1 radians for the rotation and a standard deviation of 2% of the measured distance for the translation.

5.2. Measurement Model

To calculate the weight of particles only the database location that is closest to the current particle is used. This means that the computation time will decrease as the particles converge around the true location of the robot, since fewer of the features in the database will need to be matched to the current image. The weight is based on the number of interest points that match between the current image and the corresponding database image ($N_{match}^{(i)}$). A candidate interest point match is considered if the lowest match value, calculated from the squared Euclidean distance between the histograms, M_1, is less than 60% of the next lowest match value M_2. This factor that was found empirically and also used in [10]. This guarantees that the interest point match will be significantly better compared to the other possibilities. No interest point is allowed to be matched against more than one other interest point. If an interest point has more than one candidate match, the match which has the lowest match value among the candidate matches is selected.

All particles that are closest to the same database point will have the same match value. To avoid drifting of the particles away from the database position, the weighting function $f_w(d)$ is applied:

$$f_w(d) = \begin{cases} exp\ \ -\frac{(d-\sigma)^2}{\tau^2} & (d > \sigma) \\ 1 & (d \leq \sigma) \end{cases} \tag{4}$$

where d is the euclidean distance between the particle and the database position. In the experiments, σ and τ were set to $2T$ where T is the minimum distance between database positions. The new weight is then calculated as $\pi_t^{(i)} = f_w(d) \cdot N_{match}^{(i)}$.

6. EXPERIMENTS

6.1. Building the database

The localization system consists of a database of features where one set of features is stored for each database position. The features were calculated from images taken at the known positions. To obtain these positions and the ground truth data for performance evaluation, a SLAM implementation was applied using the technique described in [11]. A total of 603 images were collected covering an area of approximately 60×55 meters, as shown in figure (5). New laser scans and images were recorded if the rotation since the previous image exceeded 15 degrees or when the translation exceeded 0.5 meters. For each image the corresponding pose estimate from the SLAM algorithm was stored. We refer to this set of images as the exploration set.

Since all the features used are rotationally invariant, it is only necessary to use images with different locations and not orientations, i.e., when the robot is travelling back and forward along a corridor it is sufficient to save the data in one direction. The building of the database starts after the run is completed and optimized with SLAM. The images are used in the same order as they were taken. An image is added to the database if the metric distance to the nearest stored image exceeds a threshold T. In this paper, a value of $T = 0.4$ meters was used. For each image included in the database, a feature set is calculated and stored.

6.2. Evaluation Method

Another set of 300 images was collected, referred to as the localization set. This set was collected over a period of two months at different times after the exploration set, resulting in changes in the environment as well as illumination. The localization set is taken mainly from the Ph.D corridor, see figure (5), which contains a lot of similar images, e.g., doors of office rooms, and a lack of furniture or objects. All of the recorded image data covers a real environment where people are moving and occluding some details

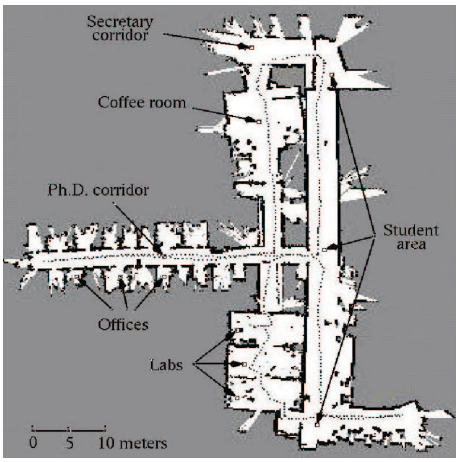

Fig. 5. Area covered by the exploration set of images.

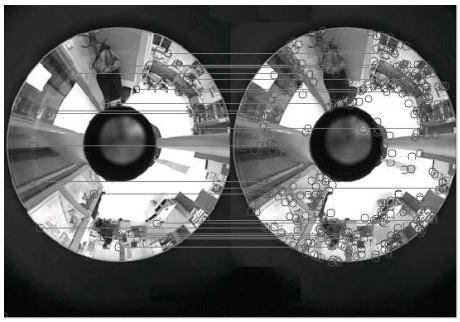

Fig. 6. Matching two different panoramic images using the proposed approach. Only a small number of features are required to get the same matching image.

which could have helped the robot to localize correctly. Other challenges also arose from the fact that many different features were detected depending on whether doors were open or closed. To obtain more evaluation data from the localization set, the experiments of the robot localization problem were repeated many times. Each time, the robots chose a new scenario of navigation using a different starting position.

When applying iterative SIFT to the robot localization problem, we try to reduce the computational effort of the feature extraction as much as possible while maintaining high localization accuracy. This means that the robot will try to localize itself using less keypoints than with classical SIFT. The keypoints in iterative SIFT are found through a random process. This makes it dangerous that the keypoints found in the exploration phase are different from those found in the localization phase, or that the common keypoints between the two are not sufficient for localization. Since the computation time of the exploration phase is not critical, we overcome this problem by applying classical SIFT in the exploration phase and iterative SIFT in the localization phase as seen in figure (6).

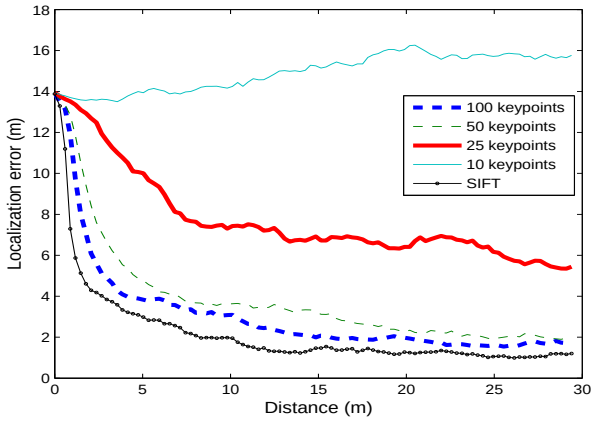

Fig. 7. Localization error against distance travelled by the robot using different number of keypoints. $NTrials = 7$ in all the experiments.

6.3. Results

We made 5 different experiments, where each experiment involves a new navigation scenario for each image in the test run, and the average results of scenarios are plotted as seen in figure (7). This means we have tested the localization process with 300 different scenarios. The figure illustrates how the robot localization process converges using the particle filter as the robot travels in a 30 meter long path. The results, when using the SIFT approach, show earlier localization convergence and higher accuracy than the others. We should not forget that this experiment needs much more computation power than the others, as seen in figure (4). When performing similar experiments with the iterative SIFT approach using different number of keypoints, the results, seen in the same figure, show that we can solve the robot localization with much less computation time and still maintaining good accuracy. For example, when using iterative SIFT with only 100 keypoints, the accuracy is only 0.7 meter worse than SIFT after the robot moves 15 meters. We can notice that the robot can localize itself with only 25 keypoints, but it needs more time than that of the experiments with more keypoints. The case of 10 keypoints shows that the robot cannot localize itself with such a small number of features.

One important parameter to adjust in the iterative SIFT approach is the number of iterations, $NTrials$, which is discussed in section 3. On one hand, reducing the number of iterations makes it possible that the search for the keypoints would be stopped at an early stage. On the other hand, increasing the number of iterations would slow down the algorithm to a high degree. In figure (8), we assign the required number of keypoints $NKeys$ to 100. The results show different accuracies using different number of iterations. We can see that using $NTrials = 7$ or $NTrials = 4$ leads to the higher accuracy than the others. Note that using $NTrials = 1$ would make the search of the keypoints fail too early, and consequently leads to unsuccessful localization. Similar results were obtained when repeating the experiment using 50 and 25 keypoints.

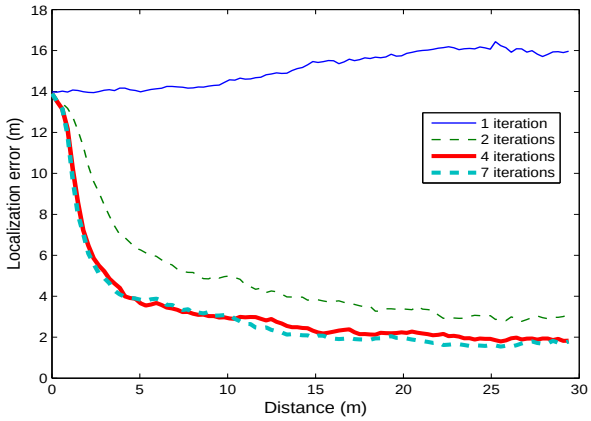

Fig. 8. Localization error against distance travelled by the robot using 100 keypoints and different number of iterations ($Ntrials$).

7. CONCLUSION

In this paper we introduced a practical idea to speed up the SIFT approach. The number of keypoints can be defined in advance and the computation time is proportional to that number. When applying the approach to the robot localization problem, we demonstrated that this approach is suitable since not many keypoints are needed. The approach can be generally applied to any similar problem. It should be obvious that any further optimization to the original SIFT approach, such as in the keypoint descriptor or orientation assignment, may also be applied to this approach. Also, any optimization to the keypoint matching process, such as using best-bin-first search, would generally have the same relative improvement on our approach as on the original SIFT approach.

Acknowledgment

The authors would like to thank Prof. David Lowe for providing them with the source code of SIFT features. The first author would also like to acknowledge the financial support by the German Academic Exchange Service (DAAD) of his Ph.D. scholarship at the University of Tübingen.

8. REFERENCES

[1] S. Se, D. Lowe, and J. Little, "Vision-based mobile robot localization and mapping using scale-invariant features," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2001*, Seoul, Korea, May 2001, pp. 2051–2058.

[2] H. Andreasson, A. Treptow, and T. Duckett, "Localization for mobile robots using panoramic vision, local features and particle filter," in *IEEE International Conference on Robotics and Automation (ICRA 2005)*, Barcelona, Spain, 2005.

[3] R. Sim and G. Dudek, "Learning generative models of invariant features," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan, 2004, pp. 3481–3488.

[4] J. Kosecka and F. Li, "Vision based topological Markov localization," in *IEEE International Conference on Robotics and Automation (ICRA 2004)*, New Orleans, USA, 2004, pp. 1481–1486.

[5] C. Silpa-Anan R. Hartley, "Localisation using an image-map," in *Proceedings of the 2004 Australasian Conference on Robotics and Automation*, Canberra, Australia, 2004.

[6] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[7] M. Artač and A. Leonardis, "Outdoor mobile robot localisation using global and local features," in *Computer vision - CVWW '04 : proceedings of the 9 th Computer Vision Winter Workshop*, Danijel Skočaj, Ed., Piran, February 2004, pp. 175–184, Slovenian Pattern Recognition Society.

[8] T. Lindeberg, *Scale-Space Theory in Computer Vision*, Kluwer Academic Publishers, Norwell, MA, USA, 1994.

[9] A. Doucet, N. de Freitas and N. Gordon, Ed., *Sequential Monte Carlo Methods in Practice*, Springer, New York, 2001.

[10] J. Gonzalez-Barbosa and S. Lacroix, "Rover localization in natural environments by indexing panoramic images," in *Proceedings of the International Conference on Robotics and Automation*. IEEE, 2002, pp. 1365–1370.

[11] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localisation and mapping," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 196–207, April 2005.