2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Localization of Mobile Robots Using Incremental Local Maps

René Iser, Arthur Martens, and Friedrich M. Wahl

Abstract— **This paper presents an algorithm for the global localization of mobile robots. The general idea is to build a local map incrementally which is matched to a global map in each localization step. The matching algorithm is a very time and memory efficient enhancement of the common Random Sample Consensus (RANSAC). It is executed a fixed number of iterations computing a set of hypotheses of the current robot pose. This paper describes how to handle the resulting hypotheses, i.e. a method is introduced deciding when the robot is localized reliably enough. The algorithm has been implemented and its characteristics are evaluated and discussed in an experimental section.**

I. INTRODUCTION

Self-localization adresses the problem of estimating the position and orientation of a mobile robot with respect to the coordinate systems of an a-priori known map of its environment. This capability is essential for most applications since safe navigation including path planning requires knowledge of the current location of a mobile robot. This problem has received considerable attention over the last decades [2], [16]. It can be decomposed into local and global localization. Local techniques aim to correct odometry errors ocurring during robot motion. In this case, a rough estimation of its global pose is available and thus the localization problem is reduced to tracking of the pose. Hence, localization on a local level is mandatory for algorithms dealing with the Simultaneous Localization and Mapping (SLAM)-problem [4] where the afore mentioned odometry errors have to be compensated.

In contrast to this, global techniques are able to localize a robot without any prior knowledge about its position or orientation w.r.t a global frame. This problem is also refered to as *kidnapped robot* problem [11] i.e., a robot disappears from its current pose and is carried to an arbitrary unknown place. A very popular approach to cope with this problem is Monte Carlo localization [16]. This approach uses a special representation of the probability density function of the Markov localization method [2]. In principle, this method draws a set of samples by random. Then, each sample is weighted according to an observation model $p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{m})$. The latter expression represents the likelihood of an observation $\boldsymbol{z}$ given the robot pose $\boldsymbol{x}$ and the global map $\boldsymbol{m}$. Afterwards, a resampling operation is performed. If these steps are iterated over time, the samples will concentrate densely around the most likely robot poses. The major drawback of Monte Carlo localization is a divergence of the filter due to an observation model specified too optimistically. This problem is also called the particle depletion problem and often occurs in highly cluttered environments where small changes of the robot pose result in large changes of the sensor readings [9].

A different algorithm which has been applied for solving the global localization problem in the past is the Random Sample Consensus (RANSAC) approach [1] originally proposed for matching arbitrary point clouds. It randomly generates a set of hypotheses by matching a minimal portion of both point sets and scoring each hypothesis by counting the number of points which could be matched. Lowe *et al.* [11], [12] use a camera system for generating a 3D map of the workspace of a robot. The 3D points and their corresponding SIFT features are stored in a database. RANSAC is now used for the localization of the robot while the SIFT features help to preselect potential matches and thus make the technique computationally feasible.

Of course, matching only one single observation to the global map is often insufficient for a reliable localization because of structural ambiguity. Tanaka *et al.* [14], [15], [17] incrementally generate a local map during localization in large size environments. The approach is termed iRANSAC and at each iteration the local map is matched to the global one by employing the RANSAC approach. The map is built by a laser range-finder and features are extracted from each scan which are used for RANSAC matching. Additionally, the authors propose different rules for choosing the local features to be matched. In [10] the approach is improved further by integrating locality sensitive hashing (LSH) [3] into the algorithm.

In this paper we propose a localization algorithm which is based on the Random Sample Matching (RANSAM) approach introduced by Winkelbach *et al.* [21]. This technique exploits the theory of the so-called birthday attack which is known from cryptography [18] and determines an upper bound for the probability of a collision in a hash table. The algorithm is very time and memory efficient and easy to implement. In the localization context it is employed for matching an incrementally built local map to a global one. The main contribution of this paper is to illustrate how to handle the hypotheses computed by the RANSAM approach which includes identifying when the robot is localized with a sufficient reliability. The great advantage of the proposed algorithm is that it does not suffer from any divergence. The remainder of this paper is organized as follows. For readers' convenience, in Section II we will briefly review the original version of the RANSAM method. In Section III the localization framework is explained in detail. In Section IV intensive experiments demonstrate the characteristics of

The authors are with the Institut für Robotik und Prozessinformatik, Technische Universität Braunschweig, 38106 Braunschweig, Germany. {r.iser,arthur.martens,f.wahl}@tu-bs.de

our localization method. Finally, Section V concludes the paper.

II. BRIEF REVIEW OF THE RANSAM ALGORITHM

In the following we will give a brief summary of the algorithm. For further information please consult [21].

Let $A = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_m\}$ and $B = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_n\}$ be two arbitrary point sets (represented by kd-trees) given w.r.t. a world frame $\mathcal{W}$. The matching operation works as follows. First, three points $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k \in A, i, j, k \in [1, m]$ are sampled randomly. These points form a triangle and define a 3D relation $rel(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k)$ determined by the side lengths of the triangle. This relation allows the points to be stored in a three dimensional hash table denoted by R_A:

$$R_A[rel(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k)] = (\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k) \tag{1}$$

Subsequently, the same process is repeated for set B: Three points $\boldsymbol{y}_p, \boldsymbol{y}_q, \boldsymbol{y}_r \in B, p, q, r \in [1, n]$ are drawn randomly and stored in a second hash table R_B: $R_B[rel(\boldsymbol{y}_p, \boldsymbol{y}_q, \boldsymbol{y}_r)] = (\boldsymbol{y}_p, \boldsymbol{y}_q, \boldsymbol{y}_r)$. This process is alternatingly repeated until a collision occurs, i.e.

$$rel(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k) = rel(\boldsymbol{y}_p, \boldsymbol{y}_q, \boldsymbol{y}_r) \tag{2}$$

In this case a pose hypothesis to compute the relative transformation between A and B is obtained. The runtime complexity to compute the first hypothesis is $O(h)$ where h is the size of the hash table (assuming rel to be unique given a concrete point triple, we only need to process $1.2 \cdot \sqrt{h}$ triples until a collision occurs with a probability of $p > 0.5$ [18]). This process converges to $O(1)$ because the hash table is continuously filled with new relations. A pose hypothesis ${}^{A}\boldsymbol{H}_{B}$ can be obtained by computing the centroid of each triangle defining two transformations matrices whose position vectors correspond to the centroids. The transformation from $\mathcal{W}$ to the centroid of the triangle of A is denoted by ${}^{\mathcal{W}}\boldsymbol{T}_{A}$; the transformation to the triangle frame of B is denoted by ${}^{\mathcal{W}}\boldsymbol{T}_{B}$. Now each point $\boldsymbol{x}_l \in A$ is transformed by the following equation:

$$\boldsymbol{x}_l{}' = {}^{\mathcal{W}}\boldsymbol{T}_{B} \cdot {}^{\mathcal{W}}\boldsymbol{T}_{A}{}^{-1} \cdot \boldsymbol{x}_l \tag{3}$$

After this matching operation the quality of the hypothesis has to be checked. In principle, this can be done by checking for each point of A whether the distance to its closest point in B is below a threshold ε. Hence, an appropriate measure of quality would be

$$\Omega = \frac{\sum_{i=1}^{|A|} contact_B(\boldsymbol{x}_i)}{|A|}. \tag{4}$$

If the value of Ω either exceeds a predefined threshold Ω_{max} or a maximum number of iterations is reached, the computation of hypotheses is aborted. The computation of Ω can be speeded up considerably by an extrapolation [21]. Instead of considering each point of A in every hypothesis check, a confidence interval within which the actual value of Ω lies with a probability of 0.95, i.e.,

$$\Omega \approx \frac{\sum_{i=1}^{k} contact_B(\boldsymbol{x}_i)}{k} \pm \frac{1.96}{2 \cdot \sqrt{k}} \tag{5}$$

may be beneficially employed. k is the number of randomly drawn points of set A that are considered in a single extrapolation step. If the upper bound Ω_{max} of the interval is lower than the quality of the current best match Ω_{best}, the hypothesis is discarded; otherwise further extrapolation steps are performed until either $\Omega_{max} < \Omega_{best}$ or all points of A have been considered. Note that correct hypotheses might be discarded due to the randomized structure of the extrapolation technique but this behavior does not influence the average quality of the matching result since a large number of hypotheses is checked.

III. THE LOCALIZATION FRAMEWORK

The RANSAM algorithm successfully has been employed in different applications like surface registration [21], medical robotics [19] or as part of a SLAM-algorithm [6]. It is a fundamental part of the DAVID laser scanner [20] which nowadays is well known to many research groups. In [5] the algorithm has been parallelized and theoretical investigations show how to distribute the point clouds to the different threads in an optimal manner. The parallelized variant is applied in this framework for localizing the mobile robot. Of course, matching only one single observation to the map of the workspace of the robot is insufficient in general due to structural ambiguity (cf. Fig. 1). Especially in a very featureless environment, one observation can be matched correctly to many parts of the map. Therefore, a matching algorithm employed for mobile robot localization must be embedded into a more sophisticated framework in order to be able to gather information over time. For this purpose, it is not sufficient to abort the computation of hypotheses as soon as the threshold Ω_{max} has been exceeded. Instead, it is preferable that the algorithm runs a fixed number of iterations and collects all hypotheses generated during the iterations which exceed this threshold. Afterwards, this set of hypotheses is the basis for further processing, e.g. deciding when the robot has been localized. Moreover, the robot has to build a local map of its environment during the localization process. Hence, not only a single observation is matched to the global map but the whole local map consisting of all observations made so far (cf. Fig. 1).

In this work we use a scan matching algorithm for computing the local map which already has been applied in [6] and is comparable to the one of [13]. Please note that an arbitrary SLAM algorithm can be employed for generating the local map, e.g. [8], [23]. Using a more sophisticated SLAM approach might even be preferable in environments where the robot has to travel a long way until the set of hypotheses collapses to only one unique hypothesis. There are some more requirements which have to be met by our localization algorithm. First, it has to handle multiple hypotheses, i.e. clustering hypotheses which are closely located to each other (cf. Section III-A) and weighting each cluster. In Section III-B the extraction of the correct robot pose is explained. Furthermore, the environment of a mobile robot is not static in general, e.g. tables or chairs are moved to different positions or people are sharing the workspace. This

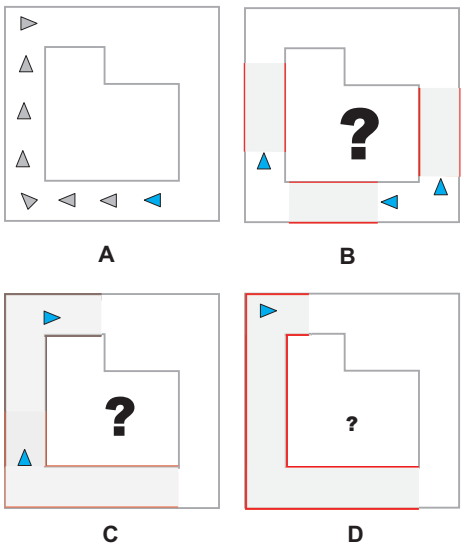

Fig. 1. A: The trajectory of the robot during the localization procedure. It starts from the pose marked by the blue triangle. B: The first observation of the robot gathered at the initial pose can be matched to many parts of the global map (highlighted by the red lines). C: The uncertainty after a few observations is decreased but nevertheless two poses have equal likelihood for the RANSAM algorithm. The pose at the bottom is the correct one in this example. D: The uncertainty of the robot pose has become small enough.

means that the quality threshold Ω_{max} may not be fixed and has to be updated permanently. Otherwise, RANSAM will not generate any hypotheses in places changed too much. A suitable update operation is introduced in Section III-C. Finally, in Section III-D a method is discussed for the detection of places which are not registered in the map. The latter aspect is important since localization becomes impossible in such regions.

A. HANDLING MULTIPLE HYPOTHESES

From a theoretical point of view there should be only one unique robot pose left when enough information has been collected. But in practical situations the RANSAM algorithm is likely to generate a set of hypotheses which are located closely to each other instead of only one single hypothesis. This is due to the fact that $\varepsilon > 0$ (cf. Section II) which allows for small translations and rotations around the *ideal* matching pose. Moreover, outliers are computed sometimes. An outlier is a hypothesis which is very unlikely but the matching quality Ω exceeds the current threshold Ω_{max} (cf. Fig. 2).

The behavior described above requires the resulting hypotheses to be clustered. To this end, we use a hierarchical K-means clustering algorithm [7], [22]. We assume that the RANSAM algorithm returns a set of hypotheses $\mathcal{H} = \left\{ {}^{A}\boldsymbol{H}_{B} \mid \Omega\left({}^{A}\boldsymbol{H}_{B}\right) \geq \Omega_{max} \right\}$. Now the K-means algorithm is initialized with one cluster $\boldsymbol{C}$ and the covariance matrix

$$\Sigma = \begin{pmatrix} \sigma_{xx}^2 & 0 & 0 \\ 0 & \sigma_{yy}^2 & 0 \\ 0 & 0 & \sigma_{rr}^2 \end{pmatrix} \tag{6}$$

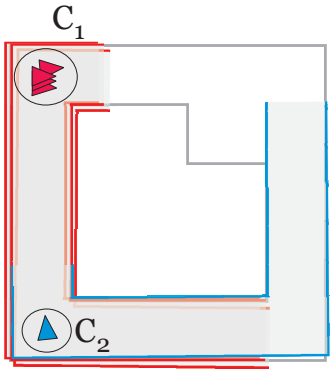

Fig. 2. Many hypotheses (red triangles) are located around the true robot pose while one outlier has been generated (blue triangle) which results from a Ω_{max} chosen small enough. E.g. in this example $\Omega_{max} \approx 0.7$ is sufficient to cause a situation like this. Furthermore, K-means computes two clusters named $\boldsymbol{C}_1$ (red triangles) and $\boldsymbol{C}_2$ (blue triangle).

of all poses belonging to this cluster is computed. For further processing, only the diagonal elements are used (cf. eq. (6)). Hence, all other elements are set to zero. The cluster is recursively subdivided if one of the diagonal elements is to large, i.e., if thresholds $\sigma_{trans,max}^2$ or $\sigma_{rot,max}^2$ are exceeded. The result is a set of clusters $\mathcal{C} = \{\boldsymbol{C}_i \mid i \in [1, M]\}$ with $M \leq \mid \mathcal{H} \mid$. One example is depicted in Fig. 2. Hypotheses marked by the red triangles are located densely around each other while one outlier (blue triangle) has been generated. Now K-means produces a set consisting of two clusters $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$. Here $\boldsymbol{C}_1$ contains all red poses and $\boldsymbol{C}_2$ contains the single blue pose. Finally, a weight is assigned to each cluster which is employed as a measure of how reliable the cluster represents the current true robot pose:

$$\begin{aligned} r\left(\boldsymbol{C}_i\right) = \alpha_1 \frac{\mid \boldsymbol{C}_i \mid}{\mid \mathcal{H} \mid} + \alpha_2 \left(1 - \frac{max\left\{\sigma_{xx,i}^2, \sigma_{yy,i}^2\right\}}{\sigma_{trans,max}^2}\right) \\ + \alpha_3 \left(1 - \frac{\sigma_{rr,i}^2}{\sigma_{rot,max}^2}\right) + \alpha_4 \frac{1}{\mid \boldsymbol{C}_i \mid} \sum_{j=1}^{\mid \boldsymbol{C}_i \mid} \Omega\left(c_{j,i}\right) \end{aligned} \tag{7}$$

with $\sum_{k=1}^{4} \alpha_k = 1$ and $r\left(\boldsymbol{C}_i\right) \in]0; 1]$. The first term of this equation rates how many hypotheses are assigned to the i-th cluster w.r.t all hypotheses generated in the current localization step. Hence, the value of the first term of the cluster $\boldsymbol{C}_2$ (cf. Fig. 2) would be much smaller than the first term of the cluster $\boldsymbol{C}_1$. The next two terms are assigned high values if the translational and rotational variances of the cluster are small. Finally, the last expression represents the average matching quality of the cluster.

B. DETECTION OF THE CORRECT POSE

Eventually, one could assume the robot to be localized if the weight of one of the clusters is high enough but in general the rating of eq. (7) is not sufficient for a reliable pose detection. E.g. in Fig. 1(C) the clusters representing the two robot poses will receive equal weights. Additionally, each weight will be very high because only two hypotheses are generated by the RANSAM algorithm in this example.

Hence, choosing the cluster with the highest weight will cause a wrong localization with a probability of fifty percent. Thus, another rating $\overline{r(\boldsymbol{C}_i)}$ is introduced which is defined as

$$\overline{r(\boldsymbol{C}_i)} = \frac{r(\boldsymbol{C}_i)}{\sum_{j=1}^{|\mathcal{C}|} r(\boldsymbol{C}_j)}. \tag{8}$$

The rating $\overline{r(\boldsymbol{C}_i)}$ of a cluster $\boldsymbol{C}_i$ will only get a significant value if $r(\boldsymbol{C}_i)$ is considerably higher than the ratings of all other clusters and thus ambiguous situations can be detected much more reliably. E.g. in Fig. 1(C) $\overline{r(\boldsymbol{C}_1)} \approx \overline{r(\boldsymbol{C}_2)} \approx 0.5$ while in Fig. 2 $\overline{r(\boldsymbol{C}_1)} >> \overline{r(\boldsymbol{C}_2)}$ and $\overline{r(\boldsymbol{C}_1)} >> 0.5$. The robot is assumed to be localized when good clusters are found exceeding a threshold in several consecutive localization steps.

C. UPDATING THE MATCHING THRESHOLD

So far, only ideal situations have been considered. More precisely, we assumed that the environment of the robot is static, i.e. the map represents the real workspace correctly and no human beings share the workspace with the robot. Of course, this is not the case in reality. Furniture like tables or chairs are often moved to different places and after some time the discrepancy of some parts of the map and the real workspace is significant. As a consequence, if a fixed Ω_{max} is used, the matching algorithm will not be able to provide hypotheses if the local environment has changed too much. Thus, the threshold Ω_{max} should be adapted permanently to the current situation.

$$\Omega_{max}(T) = \frac{1}{\tau} \sum_{t=T-\tau}^{T} \frac{1}{\mid \mathcal{H}(t) \mid} \sum_{j=1}^{|\mathcal{H}(t)|} \Omega(\boldsymbol{H}_j)(t) \tag{9}$$

The robot has performed T localization steps and the current Ω_{max} is calculated as the average of the average quality of all hypotheses computed in one localization step. In eq. (9) $\mathcal{H}(t)$ is the set of all hypotheses of the t-th localization step and $\Omega(\boldsymbol{H}_j)(t)$ is the matching quality of the j-th hypotheses of localization step t. The parameter τ determining the size of the sliding window has to be chosen carefully. If τ is small, the new matching threshold is very sensitive to environmental changes. Conversely, if τ is too large, Ω_{max} adapts only slowly to the current situation.

D. DETECTION OF UNKNOWN PLACES

Another aspect of mobile robot localization is the detection of unknown places. This means that the robot could enter areas of its environment which are not registered in the global map and thus localization becomes impossible in such regions. E.g. such a situation might occur if a door has been opened shortly before localization which was closed during the mapping process. One way to deal with this problem is to check how many points of the last τ' observations are in contact with a point of the global map w.r.t. the best hypothesis found in the current localization step:

$$\begin{aligned} \mathcal{S} &= \{\boldsymbol{S}_{T-\tau'}, \boldsymbol{S}_{T-\tau'+1}, ..., \boldsymbol{S}_T\} \\ \boldsymbol{H}_{best}(T) &= \max_{\Omega} \{\boldsymbol{H} \mid \boldsymbol{H} \in \mathcal{H}(T)\} \\ \Omega'(T) &= \frac{1}{\mid \mathcal{S} \mid} \sum_{i=1}^{|\mathcal{S}|} \frac{1}{\mid \boldsymbol{S}_i \mid} \sum_{j=1}^{|\boldsymbol{S}_i|} contact_{Map}\left(\boldsymbol{y}_{j,i}\right) \\ \boldsymbol{y}_{j,i} &= \boldsymbol{H}_{best}(T) \cdot \boldsymbol{x}_{j,i} \end{aligned} \tag{10}$$

The set $\mathcal{S}$ represents the last τ' observations and $\boldsymbol{S}_k, k \in [T-\tau', T]$ is the set of points registered to the local map during the k-th observation. $\boldsymbol{x}_{j,i}$ is the j-th point of the local map of observation i which is transformed to the global map by the current best hypothesis $\boldsymbol{H}_{best}(T)$. The idea of this approach is that it is very unlikely that more than a few points of an unknown region are in contact with points of the global map because there does not exist any point-to-point correspondence. Of course, this method may fail if the robot is located in a place which looks similar to another region registered in the global map.

IV. EXPERIMENTAL RESULTS

We used a system equipped with an Intel Core 2 Quad Q9300 "Yorkfield" processor (256KBytes L1-cache, 6144KBytes L2-cache) running at $2500MHz$, an ASUS P5Q-E motherboard, and 2 GB of RAM with a clock rate of $1066MHz$ for testing our localization algorithm. Moreover, we employed a Windows XP OS with a Visual Studio 2003 compiler. All real sensor data has been gathered by a Sick LMS 200 laser range-finder with a field of view of 180° and a resolution of 0.5°. In the following we discuss both simulated results as well as results from a real scenario. The robot we used to carry out the real world experiment was a meccanum wheel omnidirectional drive vehicle.

The RANSAM algorithm described in Section II has been configured as follows. The number of iterations per matching operation was set to $200,000$ and the contact epsilon was set to ε=1.0 (cf. eq. (4)). The choice of ε strongly depends on the geometrical structure of the point clouds to be matched. The closer the points are located to each other the smaller ε can be chosen. The quality threshold Ω_{max} was initialized differently in both experiments. The reason is that the simulation environment is a perfect world and thus more points will be in contact compared to the real world scenario where many things have changed since the global map has been generated. Furthermore, the size of the sliding window for updating the matching threshold needs to be defined (cf. eq. (9)). Empirical investigations have shown that τ=10 yields good results. Again, if τ is too small, the algorithm is very sensitive to small environmental changes while τ chosen too large results in a very slow adaption to the current situation. The queue size for the detection of unknown regions (cf. eq. (10)) also was set to τ'=10. The maximum translational variance until splitting a cluster was set to $\sigma^2_{trans,max}$=30 cm^2 and the maximal rotational variance was set to $\sigma^2_{rot,max}$=50°^{2}. Finally, $\overline{r(\boldsymbol{C}_i)}$ had to be larger than 0.8 for at least three times. In both environments,

RANSAM does not provide any clusters from time to time. In this case, the localization did not fail. The counter which has been incremented when $\overline{r(\boldsymbol{C}_i)} > 0.8$ was simply reset.

A. Simulation Results

The map of the simulated environment can be seen in Fig. 4 (a). The environment has an extension of 26.16 m $\times$ 26.7 m and the corresponding map has a resolution of 3cm. The area looks quite simple but from a localization point of view it is not easy to determine the correct robot pose due to many structural ambiguities, e.g. there are only bare walls without any distinctive features like tables, chairs, or doors.

The robot starts the localization at the pose marked by the green triangle as depicted in Fig. 4 (b). The set of clusters resulting from the first matching operation is illustrated in Fig. 4 (a). In this experiment a Sick laser with a maximum range of 8 m as employed in the real world scenario is simulated. Thus, a large portion of the scan points belong to the corridor in front of the robot while only a few scan points can be assigned to the corridor branching to the right of the robot. Consequently, almost every position along the corridors could be the real robot pose. Fig. 3(f) depicts the number of clusters plotted against the localization steps. As can be verified, the first trial yields almost 800 clusters. Thus, the matching time is still very high (cf. Fig. 3(a)). RANSAM is iterated 200,000 times but during these iterations many successful hypotheses are generated which have to be saved. In later steps, when the robot pose can be more isolated, the hypotheses management is no longer a significant part of the matching operations and the matching time collapses. One could argue that the complexity of the local map matching increases with every localization step because the size of the local map grows permanently. In principle, this is correct but Fig. 3 (a) clearly shows that the matching time per localization step increases only very slightly. The reason is that the computation of the matching quality Ω (cf. Section II) employs an extrapolation which is aborted very early most of the time as soon as a good matching has been found. This is due to the fact that if the robot pose becomes unique a very good point to point correspondence needs to be found to be considered as a good matching and this is a relatively rare event. Hence, point sampling and collision detection require a significant part of the computation time at this stage of localization. Since the number of iterations is fixed, the processing time remains approximately constant.

The next interesting area entered by the robot is the one highlighted by the blue triangle of Fig. 4 (b). At this point the robot is still not localized. The current area surrounding the robot is completely unknown, i.e. it is not registered in the map. As a consequence, Ω' continuously decreases to almost zero (cf. Fig. 3(d)). This curve shows that the computation of Ω' is a good indicator for the detection of unknown areas. Note that the best matching result per iteration Ω_{best} and Ω_{max} reach their minimum *after* the unknown area has been left by the robot (cf. Fig. 3 (b) and Fig. 3 (c)). The reason is that (almost) no hypotheses are generated by RANSAM while the robot is located within this part because no point of each new observation has a corresponding point in the global map. Hence, it is not very likely that Ω_{max} is exceeded. Since hypotheses are generated very rarely, the quality threshold is updated very slowly. As soon as the robot reenters a known region, hypotheses easily can be generated again. The robot is localized at the pose highlighted by the red triangle of Fig. 4 (b). In this scenario, 80 steps are necessary for a reliable localization. The point of time at which the robot has been localized, is marked by the black line in Fig. 3(f). The localization error is depicted in Fig. 3(e). The maximum translational error is 8.34 cm and the maximum rotational error is 0.18°. The reason for these small errors is that this *ideal* world has walls which have a thickness of only 1 grid cell and hence an acceptable matching result is close to the optimal one simultaneously. After the first successful localization we let the robot explore the whole map which required a total of 136 steps resulting in the statistics of Fig. 3. Fig. 4(c) shows an example were our localization scheme fails because the map is almost symmetric and thus at least two clusters with the same weight will be computed most of the time.

B. Robotics Lab

The second experiment took place in our robotics lab. The global map has a resolution of 3 cm and it is depicted in Fig. 5 (a). Our lab has a size of 22 m $\times$ 14 m. The initial robot pose is highlighted by the green triangle in Fig. 5 (c). The first laser scan results in the clusters depicted in Fig. 5 (a). Again, one observation is not sufficient for localizing the robot. The first matching operation only causes 19 clusters (cf. Fig. 3(f)) which is a very small number compared to the simulated environment. Consequently, the matching time does not exhibit such a strong decrease but remains approximately constant all the time (cf. Fig. 3(a)). The robot is localized for the first time at the red triangle of Fig. 5 (c). Again, this point is marked by the black line in Fig. 3 (f). The largest translational error is 13.16 cm and the largest rotational error is 2.12° (cf. Fig. 3 (e)). Although the robot has been localized correctly at this point, we still let it explore the unknown region marked by the blue triangle of Fig. 5 (c) in order to show that the characteristics of the algorithm are the same in reality. Fig. 5 (b) shows the matching of the complete local map to the global one. Of

course, all curves oscillate much stronger than the curves of the simulation because this map represents a very cluttered environment compared to the *ideal* world of the simulation. Moreover, many things have changed in detail since the map has been generated causing the relatively strong oscillation of Ω'.

V. CONCLUSION AND FUTURE WORK

In this paper we presented a localization method for mobile robots based on a very fast matching algorithm for 3D point clouds. During localization the robot computes a local map of its environment which is matched to a global map. The local map is generated using a simple scan matcher which easily can be replaced by a more sophisticated SLAM approach. The result of matching the local map to the global one is a set of hypotheses of potential locations. Afterwards, these hypotheses are clustered and weighted. In this manner, the algorithm decides when the robot is localized reliably enough. Except from an incrementally growing local map no information from previous localization steps are exploited and thus our method does not suffer from divergence. Moreover, we described how to detect places which are not registered in the global map based on the matching result. Hence, our approach is able to prevent the robot from exploring areas which do not give any additional information concerning localization. The characteristics of our algorithm have been evaluated experimentally.

In Section IV-A an example is given where our localization method will fail. In the future, we plan to improve our concept in order to make it robust against such situations. Moreover, future research will focus on some aspects of the local map, e.g. if the extension of the local map is restricted, the computation time per localization step would be constant which is very important for real time applications. Since we stated that our algorithm does not suffer from filter divergence, a comparison with traditional Monte Carlo Localization is essential, too.

VI. ACKNOWLEDGMENTS

We would like to thank *QNX Software Systems* for providing free software licenses. Furthermore, the authors would like to thank Daniel Kubus and Markus Rilk for their helpful comments on the draft of this paper.

REFERENCES

[1] A. M. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 46(6):381–395, 1981.

[2] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Artificial Intelligence Research*, 11:391–427, 1999.

[3] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. of the 25th Very Large Database Conference (VLDB)*, pages 718–728, 1999.

[4] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. on Robotics*, 23(1):364–375, 2007.

[5] R. Iser, D. Kubus, and F. Wahl. An efficent parallel approach to random sample matching (pRANSAM). In *Proc. of IEEE International Conference on Robotics and Automation*, pages 1199–1206, 2009.

[6] R. Iser and F. Wahl. Building local metrical and global topological maps using efficient scan matching approaches. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1023–1030, 2008.

[7] S. Lamrous and M. Taileb. Divisive hierarchical k-means. In *Proc. of the IEEE International Conference on Computational Intelligence for Modelling Control and Automation*, pages 18–18, 2006.

[8] J. Nieto, T. Baily, and E. Nebot. Recursive scan matching slam. *Journal of Robotics and Autonomous Systems*, 55(1):39–49, 2007.

[9] P. Pfaff, C. Stachniss, C. Plagemann, and W. Burgard. Efficiently learning high-dimensional observation models for monte-carlo localization using gaussian mixtures. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3539–3544, 2008.

[10] K. Saeki, K. Tanaka, and T. Ueda. LSH-RASNAC: An incremental scheme for scalable localization. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 3523–3530, 2009.

[11] S. Se, G. D. Lowe, and J. J. Little. Global localization using distinctive visual features. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 226–231, 2002.

[12] S. Se, G. D. Lowe, and J. J. Little. Vision based localization and mapping for mobile robots. *IEEE Trans. on Robotics*, 21(3):364–375, 2005.

[13] D. Silver, D. Bradley, and S. Tayer. Scan matching for flooded subterranean voids. In *IEEE Conference on Robotics Automation and Mechatronics*, pages 422–427, 2004.

[14] K. Tanaka and E. Kondo. Incremental ransac for online vehicle relocation in large dynamic environments. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 1025–1030, 2006.

[15] K. Tanaka and E. Kondo. Towards constant-time robot localization in large dynamic environments. In *Proc. of IEEE Int. Conf. on Networking, Sensing and Control*, pages 113–118, 2006.

[16] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence Journal*, 128(1-2), 2001.

[17] T. Ueda and K. Tanaka. On the scalability of robot localization using high-dimensional features. In *Proc. of the Int. Conf. on Pattern Recognition*, pages 1–4, 2008.

[18] E. W. Weisstein. Birthday attack. from mathworld - a wolfram web resource. ⟨http://mathworld.wolfram.com/birthdayattack.html⟩ [date: 07/08/2009]. Internet, 2009.

[19] R. Westphal. *Sensor-Based Surgical Robotics: Contributions to Robot Assisted Fracture Reduction*. Fortschritte in der Robotik. Shaker, 2007.

[20] S. Winkelbach and S. Molkenstruck. David 3d scanner. ⟨http://www.david-laserscanner.com⟩ [date: 07/06/2009]. Internet, 2009.

[21] S. Winkelbach, S. Molkenstruck, and F. M. Wahl. Low-cost laser range scanner and fast surface registration approach. In *Pattern Recognition (DAGM 2006)*, pages 718–728, 2006.

[22] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier. Morgan Kaufman, 2005.

[23] K. M. Wurm, C. Stachniss, G. Grisetti, and W. Burgard. Improved simultaneous localization and mapping using a dual representation of the environment. In *Proc. of the 3rd European Conference on Mobile Robots*, pages 132–137, 2007.

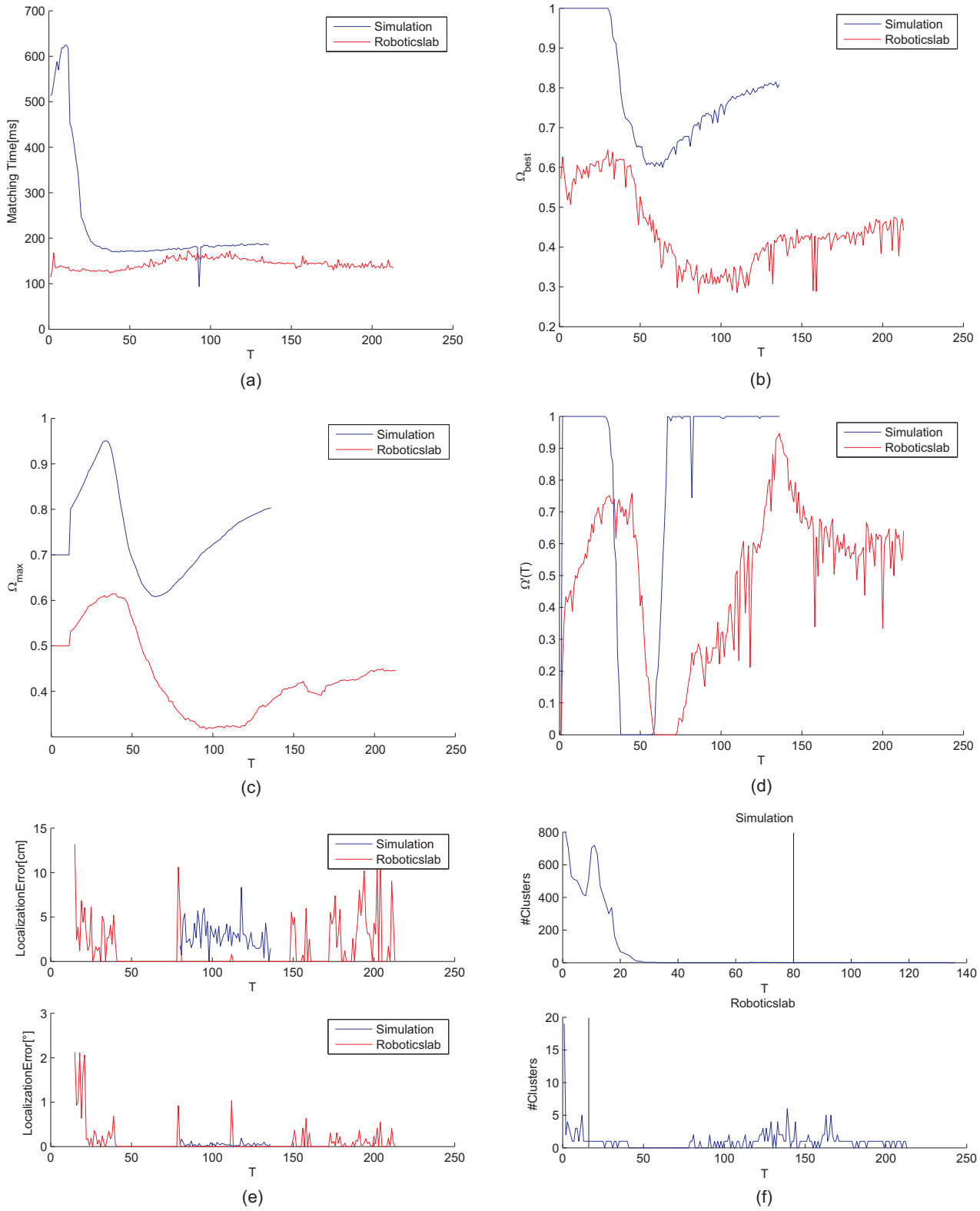

Fig. 3. The abscissas do not have a unit. They simply represent consecutive localization steps. (a) Computation time of each matching operation. (b) The best matching result Ω_{best} per iteration. (c) The matching threshold Ω_{max} is updated in every iteration. In both scenarios the threshold decreases constantly while the robot is moving in regions not registered in the global map. (d) Portion of points belonging to the last $\tau' = 10$ scans which are in contact with a corresponding point of the global map. These curves clearly depict when the robot enters and leaves unknown areas of the maps. (e) The localization error. Note that in the simulated scenario the robot could not be localized until iteration 80. (f) The number of clusters resulting from each matching operation. The point at which the robot has been localized is marked by the black lines.

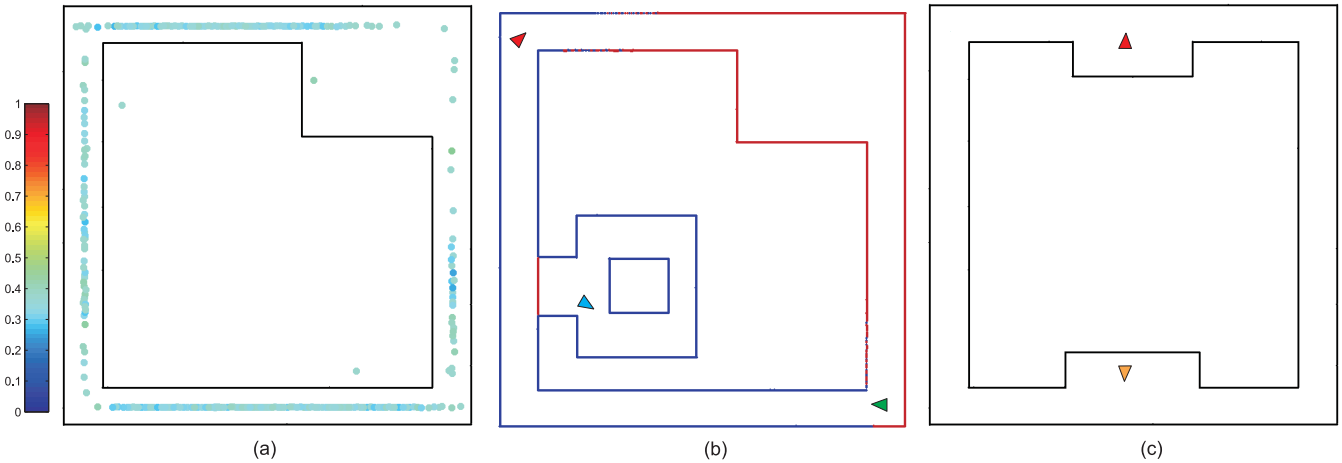

Fig. 4. (a) The global map and all clusters resulting from the first matching operation. The color represents the unnormalized weight $r(\boldsymbol{C}_i)$ of the clusters $\boldsymbol{C}_i$ according to the color bar on the left hand side. Nevertheless, $r(\boldsymbol{C}_i) \in]0;1]$ (cf. eq. (7)). The dark blue dots have the lowest weight. The higher the certainty the more turquoise are the clusters. Thus, all weights range between 0.2 and 0.45. Note that there are no yellow or even red clusters in the first iteration since all hypotheses are vastly spread over the whole workspace. (b)The local map matched to the global one at the localization point (cf. Fig 3 (f)). The initial robot pose is represented by the green triangle. The region around the blue triangle is not registered in the global map. This is a good example for a scenario in which other parts of the environment become accessible for the robot, e.g. a door has been opened. The red triangle marks the pose at which the robot has been localized. (c) An example were our algorithm fails due to the symmetry. The bulge at the bottom is slightly larger than the bulge at the top. Nevertheless, our algorithm generates at least two solutions most of the time. For example, let the true robot pose be the one marked by the red triangle. Another solution which is very likely is the one highlighted by the orange triangle.

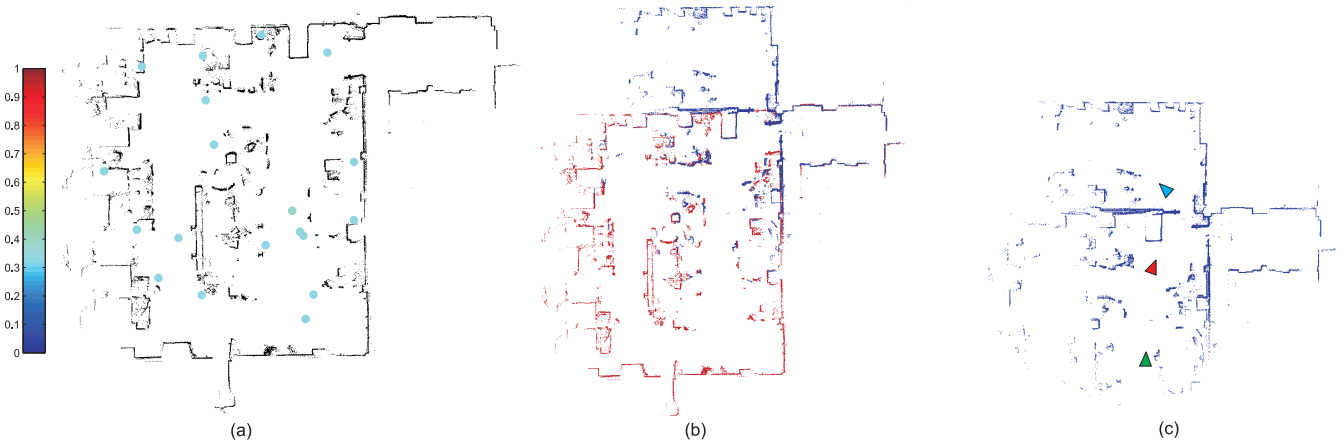

Fig. 5. (a) The map of our robotics lab. The dots represent the clusters resulting from the first matching operation and thus one scan is not enough for localizing the robot in this environment. Again, the color encodes the cluster rating in the same manner as in Fig. 4 (a). (b) The local map of (c) matched to the global map. (c) The final local map. The green triangle represents the initial robot pose. The red triangle marks the pose at which the robot has been localized. Finally, the blue triangle depicts the pose at which the robot detects that it is located in an unknown region because Ω' is almost zero (cf. Fig 3 (d)).