Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems

Active Exploration for Feature Based Global Localization

Marco Seiz
mseiz@nubix.ch
Autonomous System Laboratory,
Swiss Federal Institute of Technology,
1015 Lausanne EPFL, Switzerland.

Patric Jensfelt
patric@s3.kth.se
Henrik I. Christensen
hic@nada.kth.se
Centre for Autonomous Systems,
Royal Institute of Technology,
Stockholm SE-100 44, Sweden.

Abstract

This paper presents an algorithm for active exploration of the environment by a mobile robot when performing global localization. During the localization process interesting regions for future exploration are selected based on already detected features and on the hypotheses generated by the localization algorithm. The localization process is improved by presenting it a richer set of features. The proposed algorithm provides highly robust global localization in real world environments with very low computational effort spent in finding exploration goal points. Experimental results are given, demonstrating the effectiveness of the algorithm in a number of different situations.

1 Introduction

This paper proposes an algorithm for *active exploration* (AE) in the field of *global localization*. *Global localization* is the problem where a robot, holding a map of the environment, should determine its *position and orientation* (pose), without prior knowledge. The basic idea of this work is to improve the performance of an existing *passive global localization algorithm* (PGL) [1]. The existing PGL uses an *open space explore* behavior for exploration, driving the robot around randomly in the environment. This paper proposes a more intelligent solution, which actively moves the robot platform during the localization process to collect better information for determining the pose. With AE, localization can be achieved in more complex situations and it takes less time on average to find the correct pose. The algorithm can be applied in any system having a feature based PGL, delivering pose hypotheses and their respective weight (probability).

When only using the *open space explore* behavior to search the environment, doors are unlikely to be passed. The robot stays in the initial room and is not

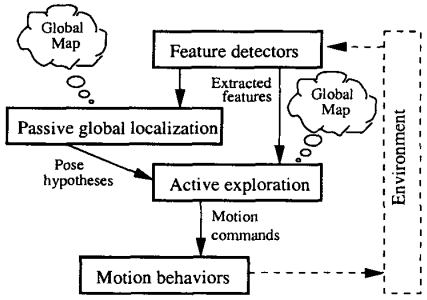

Figure 1: Closed loop exploration possible through the active exploration algorithm

able to localize when similar rooms exist in the map. The AE selects a region of interest for exploration. The selection is based on how much information a certain region contains and what the cost is to get there. Neighboring rooms are also considered and a change of room can hence be imposed. The algorithm selects the most promising region, decides how to get there and where to face the sensors. Once the region is explored a new decision is taken.

An enhanced localization performance is achieved by the indirect feedback from the AE to the PGL through the sequence of detected features. The AE does not insist on observing an expected feature, since its existence and visibility are not certain. The approach of having the AE as an independent module allows easy integration into other systems. Figure 1 shows the structure and relationships in the localization system with AE.

The PGL is responsible for delivering a set of pose hypotheses to the AE. The number of hypotheses internal to the PGL strongly depends on the localization method. During the exploration, the hypotheses are updated based on sensor data. The PGL reports that localization has been accomplished when a single pose hypothesis gets dominant enough based on its weight.

A set of the strongest hypotheses is communicated

0-7803-6348-5/00/$10.00 ©2000 IEEE.

from the PGL to the AE. To evaluate these pose hypotheses the algorithm also needs access to the global map. Observed features can be compared to the map, allowing AE to select interesting exploration targets based on not yet detected features. The algorithm's output is a sequence of motion commands for exploring the selected target regions and features.

2 Related Work

Variations of this problem have already been addressed by different authors. Dudek et al. [2] decomposed the map into visibility regions defined by polygons. The concept was extended to visibility skeleton matching in [3] and [4]. Localization is not possible in a visibility polygon unless it is unique. A precomputed path is followed to locations that can rule out hypotheses until there is only one left. This algorithm can be unstable because it assumes two things: i) when inside a visibility region, we can match this region to a set of identical regions in the map; ii) each set of regions of identical visibility polygons can be identified based on the map. If one of these assumptions is not fulfilled, the robot will take exploration decisions, which are not meaningful in reality, or it will eliminate the correct hypothesis by accident.

Burgard et al. [5] addressed the problem using a grid based approach, achieving good results by basing the decision of where to explore on minimization of the expected future entropy of the hypotheses distribution. Unfortunately the required processing power is proportional to the area and to the resolution of the grid. Our proposition is a feature-based approach, only considering the relevant part of the map when taking a decision. This eliminates the problem of being dependent of the map size and is thus less costly in processing power.

3 Algorithm: Three Stages

The AE algorithm goes through three stages during the localization process:

1. Initialization
2. Hypotheses elimination
3. Hypothesis strengthening

During the initialization stage, the robot collects information to generate a set of pose hypotheses. False hypotheses are ruled out in the elimination stage and finally the belief of the best hypothesis is improved in the strengthening stage until the robot is considered localized.

The stage of the AE is selected based on the weights of the current pose hypothesis delivered by the PGL. Thus, it is not forced to follow the sequence shown above, instead it always adapts according to the current pose hypotheses. Two thresholds are responsible for this selection. A first threshold on the total weight of the best hypotheses decides if enough information has been gathered to switch to the second stage and a second threshold decides if there is only one (but not strong enough) hypothesis left and the algorithm goes into the last stage.

Initialization The robot has no initial information regarding the pose and hence the initial hypotheses generated by the PGL are all very weak and unreliable. In this stage the exploration is controlled by the *open space explore* behavior previously used with the PGL. This behavior starts driving into the direction with the least obstacles and maintains the direction while avoiding obstacles until the path is blocked. At this point it restarts the procedure.

Hypotheses elimination Based on a set of hypotheses the robot can direct its exploration and eliminate false hypotheses by guiding the robot to discriminative locations. This leads us to the central idea of our algorithm. The decision is based on overlaying the different hypotheses and matching the features. The rooms which the robot is in, according to the hypotheses, are mapped into layers within the same robot centered coordinate system. The features are matched between these layers, and the most promising region is selected by a measure of information contents. When the differences between the rooms are small, the robot needs to change room. This decision is taken in a consistent manner using the same algorithm.

Hypothesis strengthening Only one dominant hypothesis is left, but it has not yet accumelated enough evidence. Overlaying hypotheses are no longer possible with only one hypothesis. Instead the decisions are based on the discovered features. These features are collected in a map during the entire exploration. The decision is computed as if the map containing the features were a second hypothesis. This approach makes it possible to use the same algorithm as in hypotheses elimination. Regions with unseen features and not yet visited rooms are the most interesting.

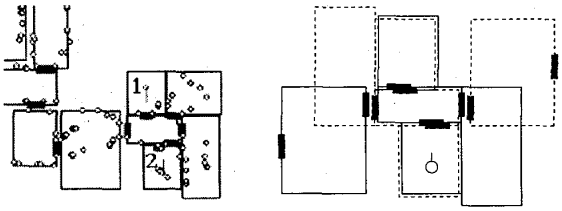

Figure 2: The left image shows a part of the map with two pose hypotheses [1,2]. The right image shows the resulting overlaid maps.

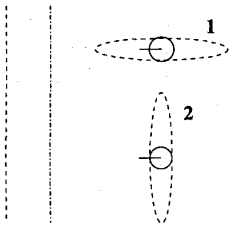

Figure 3: A robot observes two line features with different covariance on its position (1 and 2). In case 1 the lines cannot be distinguished and thus do not contain information helpful to the localization process.

4 Algorithm: Select Explore Region

The generation of a target region based on the hypotheses coming from the PGL is explained below using a top down approach.

Overlaying hypotheses The map consists of rooms which are topologically connected. The features are compared on a room by room basis to reduce the computation of the matching process. In the first place these are the rooms where the hypotheses are located. The overlaid maps are then recursively extended with rooms whenever a passable door exists in all layers at a given position (Figure 2).

Matching rooms Matching the features through all associated rooms is not done in one step but every room is compared with every other room. When matching features between two rooms every feature is matched with all features of the same type in the other room. The best match for that feature is retained and accumulated by multiplication over all room-to-room-matches as the general match, m_g, of that feature. The matching function is an exponential function where a perfect match results in 1 and no match at all is weighted 0. Thus the general match of a feature will also be between 0 and 1.

The computation time for making a decision depends on the location and number of the hypotheses. For n hypotheses, $n(n-1)/2$ pairs of rooms need to be matched.

Matching features The PGL uses different feature types. Pairs of the same feature type are matched to each other based on their parametric description. Rooms with a larger number of features require a significantly higher number of comparisons, since f features in both rooms require $f(f-1)$ comparisons. The feature matches will depend on the hypothesis (or layer) because of differing odometry covariances under which the features would be observed (see Figure 3).

A closed form covariance model [6] was implemented, this allows to efficiently compute the predicted robot pose covariance for any feature. The parameter difference involved in the match computation is reduced by the corresponding standard deviation before the match is computed.

Feature information The explore region is mainly selected by its information content but is also guided by time constraints. Our heuristic evaluating the information content i of a single feature is defined by the following product:

$$i = (1 - m_g) \cdot f_d \cdot w_h \cdot q$$

The smaller a feature's general match m_g to other features, the more information it provides. Furthermore, objects closer to the robot are more interesting, we express this with the factor f_d (explained below). Interesting features with a high probability of existence should be selected first, thus the associated weight of the hypothesis w_h (from PGL) is also taken into account. Finally to compare the information content of different feature types, we use a relative quality measure q (see Section 5).

In a first approximation the traveling time from the robot's location to a specific feature is proportional to the traveling distance. The distance d_t used in our computation has an additional component:

$$d_t = \text{path length} + \#\text{door passes} \cdot \text{door distance}$$

An artificial door distance penalizes door passing, to prevent the robot from changing room too early. The distance factor f_d is defined as:

$$f_d = 1 - \frac{d_t}{\text{search range}}$$

where the *search range* parameter is a constant value. Features with a traveling distance exceeding the *search range* will have a negative information value and will not attract any attention. In the third stage, where the detected features are treated as the second hypothesis, w_h is set to zero for that second hypothesis. This will prevent the detected features being selected as targets by the AE algorithm.

Selecting target region The detection of a feature cannot be guaranteed. Thus, it is better to select an interesting region rather than a single feature. We obtain this effect by using a Gauss filter. The filtered information i_{filt} of all features is computed at each feature location.

$$i_{filt} = \sum_{features,layers} i \cdot e^{-(d^2/2r^2)}$$

where d is the distance from current feature to other feature and r is the filter radius (in our case: 2m).

When only the filtered information is used, features in the center of the room and clusters of small features would get too much attention. We obtained good results when using i and normalized i_{filt} in equal parts.

$$i_{final} = i + i_{filt}\frac{\max(i)}{\max(i_{filt})}$$

5 Implementation

Before describing the implementation of the AE, a few words about the PGL. It is a version of the MCL algorithm [7], where the probability density function (PDF) for the robot position is represented by a set of particles. Initially, without any knowledge, these particles are spread uniformly over the environment. When updating the PDF using sensor data, hypotheses are formed when particles form clusters. The robot is supposed localized when 90% of the particles are inside a circle of 1 m.

Matching functions The PGL is based on three types of features: lines, doors extracted from laser scans and sonar triangulation points (tripoints) [8].

Lines are defined by their distance d to the robot and their orientation α. When comparing two lines we base the match on the difference of these parameters: Δd and $\Delta\alpha$.

$$match = e^{-(\frac{|\Delta d|}{c_1} + \frac{|\Delta\alpha|}{c_2})}$$

	detect	cost^{-1}	value	q
Lines	0.95	1	0.6	0.57
Doors	0.6	0.8	1	0.48
Tripoints	0.5	0.5	0.3	0.075

Table 1: Feature weight criteria

where c_1 and c_2 are selected as $c_1 = 5$ m and $c_2 = \pi/2$ rad.

Doors are matched using the distance between the center points Δd and the relative orientation $\Delta\alpha$. Where $\Delta\alpha$ is between 0 and $\pi/2$.

$$match = e^{-(\frac{|\Delta d|}{c_3} + \frac{|\Delta\alpha|}{c_4})}$$

where we use $c_3 = 2$ m, $c_4 = \pi/2$ rad.

Tripoints are compared by their relative distance Δd.

$$match = e^{-\frac{|\Delta d|}{c_5}}$$

where we use $c_5 = 5$ m.

The parameters $c_1, \ldots, c_5$ must be of similar order of magnitude, as well as being selected with respect to the PGL. The algorithm is not sensitive to these parameters, which has been verified through off-line experiments using real data.

Different feature types To be able to compare different feature types a quality measure q is introduced for each type. The quality measure takes into account several factors; probability of detection, inverse cost of detection and value of detected feature. All factors are given relative values between 0 and 1, where 1 corresponds to high quality. The quality measure is then given by the product of these factors, see Table 1.

Path planning In the beginning, when the weight of the best hypotheses is very low, we have a high risk to command the robot through a wall. Sensing the blocking wall will improve its set of hypotheses, but insisting on reaching a goal point behind that wall is pointless. To reduce the risk of issuing such commands, two heuristic limitations have been introduced: i) a new decision is taken before issuing a motion command when obstacles block the immediate path ii) the trajectory lengths is first limited to a maximum of 20 meters and then multiplied with the weight of the best hypothesis. A high tolerance is allowed in reaching a goal point: 20% of the trajectory length to allow for odometry drift and errors on the hypotheses. When the end of the trajectory is reached, a new decision is taken.

Details in the implementation of the motion commands can change from one platform to another. Motion commands should be selected to meet the requirements of the PGL. For example, probabilistic localization methods may require that the robot moves between observations of the same feature to guarantee statistical independence.

Thresholds We have two thresholds that decide the stage of the algorithm. The first is based on the sum of the weights of the best 10 hypotheses. When this total weight is above 40%, the algorithm enters the second stage and decisions are based on those hypotheses. The second threshold is the minimum weight of a pose hypothesis for being accepted. In our implementation it is set to 3%. The algorithm is not sensitive to these thresholds, but for optimal convergence they should be selected or computed according to the environment. They depend on the degree of symmetry and on the total area of the building. For example in a hospital with 400 identical rooms we will probably never reach the 40% threshold when we start inside one of those rooms. We propose two approaches to solve this problem, either we lower the threshold or we add a timeout to the algorithm.

Abandon target Obstacles or decisions based on false hypotheses can make it impossible to reach a computed goal point. In this case two problems need to be solved: i) we need to detect this failure and ii) we need to avoid repeating mistakes. We detect a failure either before the motion starts based on current sensor inputs or later by report of a motion command fail after a timeout. An avoid past function has been implemented to avoid the repetition of mistakes. The target point p_i, which lead to the motion failure is added to the set of avoid points $\mathcal{P}$. Regions around these points are made less interesting in future decisions. The information of every feature is multiplied with the filter factor f_f.

$$f_f = \prod_{p \in \mathcal{P}} \left(1 - e^{-(\frac{d^2}{2r^2})}\right)$$

where d is the distance from the feature to the center of avoid region and r is the filter radius (in our case: 1m).

6 Experiments

In this section we will experimentally show how the overall localization performance is improved when using AE. We begin by illustrating the decision process of AE through an example.

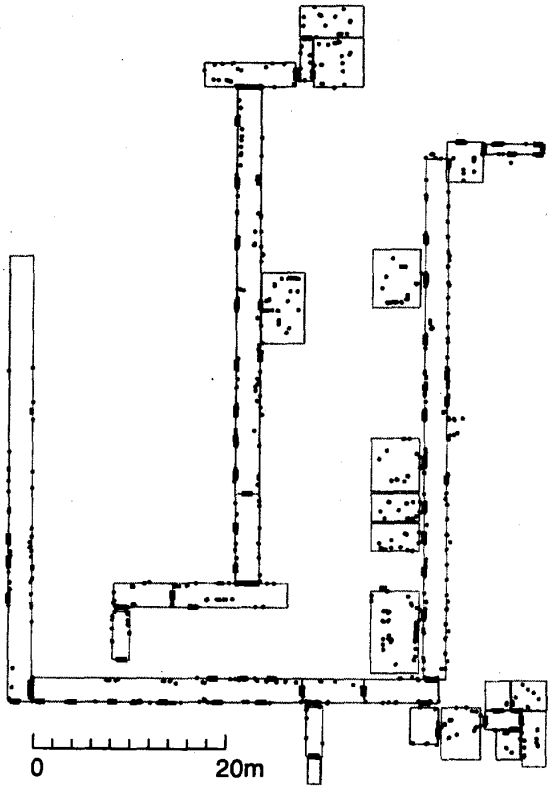

Figure 4: Map of the CAS laboratory

Shortly after starting the localization, when only a few features have been detected, there will be many weak hypotheses. Figure 5 shows the overlaid map that the decision is based on. The correspondence in one corner is good but poor in the others. At this point the map is not extended through the doors because of the large difference between the layers. After some time we are left with only two strong hypotheses, corresponding to two almost identical rooms. This was the situation introduced in Figure 2 and shown in greater detail in Figure 7.

Localization is unlikely to take place in both room **R** (see Figure 7) and in the connected hallway. The map is automatically extended through the matching doors and reveals the most relevant features two doors away in the region around the door **D**. The decision of going to the right and not to the left is based on the distance factor. On its way to the target **D**, the

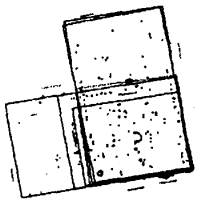

Figure 5: Overlay map with many hypotheses.

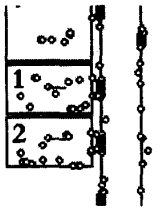

Figure 6: Experimental situation

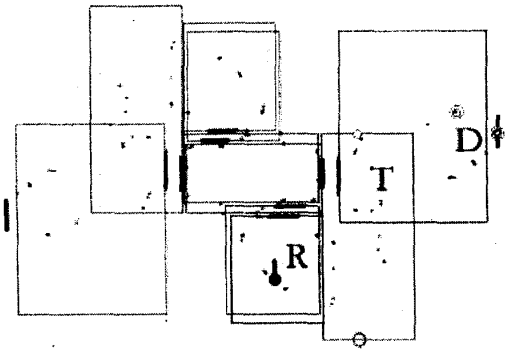

Figure 7: Overlay map with two hypotheses.

robot localizes upon receiving the first features from the target room **T**. Using only *open space explore* in this situation, would mean that it would be very unlikely that the robot would leave the first room and it would be difficult for the PGL to disambiguate the two hypotheses.

To strengthen the claim for better performance we now present a comparison over 5 real world experiments. The time is measured until the robot believes it is localized, allowing for a maximum of 10 minutes. The initial position of the robot is the same in all experiments, but the orientation is different. Figure 6 shows the experimental situation, where room 1 is the true initial room. Rooms 1 and 2 are very similar, the only difference being *tripoints*. For reliable localization it is beneficial to leave the room. Table 2 summarizes the results and it is evident that AE indeed greatly improves the performance.

Without AE the PGL did not provide a single successful localization. Instead it localized twice in the wrong room, which was not even room 2, the most similar room. This can be avoided when relevant features are observed based on the hypotheses. The same test setup, with the AE running, localized correctly in all five experiments. In all but one case the AE guided the robot through the door into the corridor. In the

Run	PGL	PGL + AE
1	4min 24s (0) - wrong	7min 54s (1) - correct
2	10min (0) - failure	3min 43s (1) - correct
3	10min (0) - failure	4min 47s (1) - correct
4	10min (0) - failure	58s (0) - correct
5	3min 34s (0) - wrong	4min 32s (1) - correct

Table 2: Time to localize in room 1, number of door traversals in brackets

Run	# Decisions	Total comp. [s]	% of exp. time
1	12	14.730	3.1%
2	4	0.954	0.4%
3	6	2.040	0.7%
4	3	0.175	0.3%
5	4	1.040	0.4%

Table 3: Computation times for experiment 1

exceptional case, run 4, the robot localized rapidly obviating the need to go out of the room.

Table 3 shows the number of decisions made in these five experiments. The total time spend for decision computation is also given in seconds and as fraction of the total time of the experiment.

On our system, running on a Pentium 550MHz, experiments have shown that the number of features per feature type and room should not exceed 200 to keep the decision taking computation below a worst case of 10 seconds.

A second series of five test was run in the so called *living room*. This room is unique in size and thus good localization performance is obtained without the AE.

As we can see from the times in Table 4, the *open space explore* performs better than AE. This is because it does not lose time for taking decisions and stopping and accelerating motions.

Run	PGL	PGL + AE
1	36s	1min 11s
2	28s	41s
3	16s	39s
4	18s	32s
5	31s	39s

Table 4: Time to localize in the living room

7 Conclusion and Further Work

Overall we can say that the exploration using the presented algorithm is very intuitive. A localization process which previously was running without feedback, now runs in a closed loop, where the loop is closed over the motion commands through the environment to the sensors.

We have been able to increase the performance of the localization process. Now even highly symmetrical situations are handled correctly and the localization process can converge rapidly. The algorithm fulfills the goal of being able to take decisions with low processing power.

One limitation of this approach is that the computation grows proportionally to the square of the number of features. A second limitation is the greedy single step search approach which cannot predict optimal search sequences. The search horizon is truncated because of the exploding computational burden and because of the limited utility of long-range planning based on noisy sensor data.

The quality measure q for features should not be constant per feature type. We expect better performance when this factor is individually set per feature. This could, for example take into account the probability, of a door being open or closed.

In some systems, including ours, we should take into account that not all rooms are in the global map. When the robot drives into such a room during exploration, that hypothesis is destroyed when the passive global localization receives features without correspondence to the map. Thus, in situations where a door does not lead to a mapped room in all hypotheses, it would be useful to save those hypotheses. Then we can later regenerate these hypotheses as soon as the robot is back in a mapped room. This requires a bi-directional communication with the passive global localization.

Acknowledgments

This fruitful collaboration was made possible by professor Roland Siegwart from the Swiss Federal Institute of Technology, Lausanne. Marco Seiz also want to thank his parents, Hugo and Silvia Seiz, who supported him financially during all his years of study. The Center for Autonomous Systems is sponsored by the Swedish Foundation for Strategic Research.

References

[1] P. Jensfelt, O. Wijk, D. Austin, and M. Andersson, "Experiments on augmenting condensation for mobile robot localization," in *Proc. of the International Conference on Robotics and Automation*, (San Francisco, CA, USA), May 2000.

[2] G. Dudek, K. Romanik, and S. Whitesides, "Localizing a robot with minimum travel," in *Proc. of the 6th annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 437–446, 1995.

[3] L. J. Guibas, R. Motwani, and P. Raghavan, "The robot localization problem," *SIAM Journal on Computing*, vol. 26, no. 4, 1995.

[4] O. Karch, "A sharper complexity bound for the robot localization problem," tech. rep., Department of Computer Science, University of Würzburg, Germany, 1996.

[5] W. Burgard, D. Fox, and S. Thrun, "Active mobile robot localization by entropy minimization," in *In Proceedings of the 2nd Euromicro Workshop on Advanced Mobile Robots*, IEEE/CS, 1997.

[6] K. S. Chong and L. Kleeman, "Accurate odometry and error modelling for a mobile robot," in *Proc. of the International Conference on Robotics and Automation*, 1997.

[7] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proc. of the International Conference on Robotics and Automation*, vol. 2, pp. 1322 – 1328, May 1999.

[8] O. Wijk, P. Jensfelt, and H. Christensen, "Triangulation based fusion of ultrasonic sensor data," in *Proc. of the International Conference on Robotics and Automation*, vol. 4, (Leuven, Belgium), pp. 3419–24, IEEE, May 1998.