Multiple Robot Navigation and Localization Using Sonar Sensors in an Indoor Environment *

Jinsuck Kim
Dept. of Computer Science
jinsuckk@cs.tamu.edu

Roger A. Pearce
Dept. of Computer Science
rap2317@cs.tamu.edu

Nancy M. Amato
Dept. of Computer Science
amato@cs.tamu.edu

Technical Report TR01-004
PARASOL LAB
Department of Computer Science
Texas A&M University
October 3, 2001

Abstract

In this paper, we describe a method for navigation and localization of formations of multiple robots in imperfectly known environments. The localizer is a modified version of a previous method for perfectly known environments which identified geometric features of the environment during preprocessing for subsequent use during localization. The new localizer handles uncertain environments or moving obstacles using new techniques for selecting subgoals along the path, one for effective measurement and the other for rearrangement of multiple subgoals to prevent the robots from blocking each other. We present results including simulation and hardware experiments in a real environment using two robots.

*This research supported in part by NSF CAREER Award CCR-9624315, NSF Grants IIS-9619850, ACI-9872126, EIA-9975018, EIA-0103742, EIA-9805823, ACI-0113971, CCR-0113974, EIA-9810937, EIA-0079874, and by the Texas Higher Education Coordinating Board grant ARP-036327-017.

1 Introduction

In the near future, manual work tasks will become more automated as technology improves. Independent service robots will prove themselves useful in diverse ways such as search and rescue, cleaning, and aiding the handicapped. These tasks are examples of potential uses of low cost mobile robots in indoor environments. The robot system can be expanded as needed by employing additional robots for the same task, which may result in increased payload, faster processing or enhanced reliability.

Some of the difficulties involved in effectively using service robots are the implementation of an automatic path planner and a robust way of dealing with odometer error. Periodic localization is necessary to reset the error, and one low cost approach is to use sonar range sensor(s). Though widely used, these sensors have several limits such as restricted range and noise. In the multiple robot case, additional complications arise such as the sensor signal may be blocked by other robots which appear as moving or unknown obstacles.

Previous work on the topic of localization includes image and landmark based localization for multiple robots [14] which requires omnidirectional vision sensors and landmarks. For one robot with only sonar sensors, there are several approaches such as probabilistic density functions [8], principal component analysis [15], Dempster's rule of combination [19], or model matching by coordinate transformation for identifying geometric features of environment [1]. Various sonar sensor configurations for effective measurements have been investigated in [4] and one of them is used for localization in [5].

Most of the previous work on multiple robot navigation has been focused on efficient collision–free path generation using techniques such as probabilistic roadmap methods(PRM) [18], on–line collision avoidance [12], a prioritized A^*–based planner [6], or a centralized optical guidance system [16].

Path planning of multiple robots under constraints is a special case. If several robots are carrying an object, the constraint is described by kinematic equations and is classified as a closed–chain constraint [11]. For this, a kinematics based probabilistic roadmap method has been proposed in [7]. Another example is the leader–followers problem where tracking control using a virtual robot and $l-l$ control has been combined to a new feedback control [9]. These approaches do not include the issue of localization.

In this work we present a system which performs navigation and localization of multiple robots using sonar sensors. We assume that a map of the environment is given. A path is extracted from a pre–computed roadmap, and geometric feature information about the environment is pre–processed for efficient localization. For multiple robot localization, path optimization techniques are discussed which not only avoid collisions, but also effectively measure the environment. This is the result of our effort to harmonize the path planner and the localizer so that they assist each other. Unlike other approaches for filtering noisy sonar sensors, we use a very simple filter that can work in conjunction with our localizer. Our framework is an extension of our previous work described in [10] where navigation and localization for one robot in a perfectly known environment is discussed.

2 Feature Based Localization

The pseudo code in Figure 1 describes our system. In a preprocessing phase, a roadmap is generated and the environment is subdivided into sectors in which localization can be easily performed. Initially, the robots are in their starting configurations and the goal configurations are known. First, a path for one robot is extracted, and we select an appropriate subgoal on that path where the robot should attempt to localize. To assist localization, the subgoal is selected which enables

localization using geometric features easily measured by the sensors. From the path, multiple paths are computed by replicating and translating segments. These "parallel" paths are useful if the robots' movements are coordinated. After this, subgoals may need to be rearranged to eliminate potential interference among the robots during localization. This is the second subgoal optimization. Note that the path generation steps are interleaved with subgoal optimization. The above is repeated until the goal is reached.

```
NAVIGATOR(start, goal)
1.  preprocess environment
2.  while goal is not reached {
3.      extract path for one robot from start to goal
4.      optimize subgoal in the path
5.      generate multiple paths
6.      rearrange subgoals in the paths
7.      drive all robots to subgoals and stop
8.      scan and localize one by one
9.      set start(s) to each robot's current configuration
10. }
```

Figure 1: Pseudo-code for our system.

2.1 Definitions

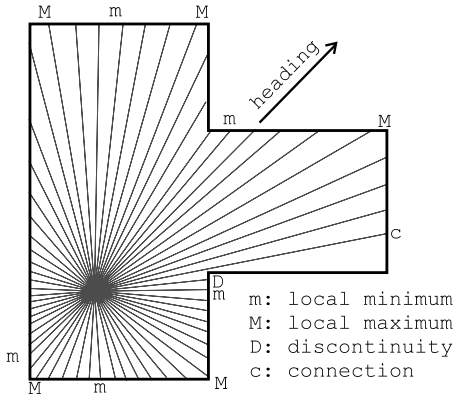

Figure 2: Simulated sensor readings and corresponding characteristic points `mMmMmMmMmDcM`

The basic building blocks of our localizer are *characteristic points* which are obtained from 360^o range scanning of the environment. Local minimum `m` (maximum `M`) points are scans where both adjacent scans are father (nearer). To determine if the scan data is discontinuous (`D` or `c`), we use a fixed threshold value for the range difference. Figure 2 shows the characteristic points of a T shaped environment.

Definition: A *feature* is a characteristic point in the scan representing a geometric feature in the environment that can be used for localization.

Though many localization techniques use the concept of a geometric feature, we are most interested in features corresponding to characteristic points in the scan. In the current implementation, only the local maximum is used as a feature since its position is fixed with respect to both the

global and the local (robot) coordinate system and provides two–dimensional information when used with adjacent characteristic points.

During preprocessing, we partition the environment into sectors so that we can quickly decide which feature to use for localization when the robot is in that sector. Our previous work used (original) visibility sectors for this purpose.

Definition: A *visibility sector* is a maximal planar region of the environment such that the same set of features is visible from all points in the sector.

In this work, we introduce a relaxed version of the visibility sector which depends only on the visibility of some common feature(s), but doesn't require all points in the sector to see the same set of features.

Definition: A *relaxed visibility sector* is a planar region of the environment such that there is at least one feature that is visible from all points in the sector.

While the subdivision induced by the visibility sector definition is unique, the relaxed sector subdivision is not unique. For example, if there are two adjacent sectors that can see a common feature, we may or may not merge them. If we merge them, then we allow the localizer to select among many different features which may be useful or inefficient, depending on the situation. To investigate these tradeoffs, we propose three desirable properties for good 'relaxed sectors'.

Sector property #1. The number of adjacent sectors which can see the same feature is small. This leads to a reduced total number of sectors (and storage).
Sector property #2. The number of features which can be scanned from a sector is large. This allows flexibility in choosing a feature during localization which increases robustness against unknown or moving obstacles.
Sector property #3. The size of a sector is not too small or too large, and in particular, the range limit of the sensor is not exceeded. Minimizing distance is also important because as the distance from a feature increases, localization is more difficult.

In general, the desired relaxed sector will maximize properties #1 and #2 while satisfying property #3.

2.2 Computing Relaxed Sectors

To compute relaxed sectors, we first compute original sectors and then merge them according to a set of heuristic rules.

If we randomly select a valid pair of sectors and merge, we lose control over the properties of the resulting sectors. As an example, consider the two cases shown in Figure 5 in which the size and shape of the relaxed sectors are very different although both were constructed from the same visibility sectors shown in Figure 3(b). In order to satisfy sector property #3, we need to keep the distance from a feature to a sector boundary as small as possible. One way of doing this is to use the Voronoi Diagram and merge sectors that are in the same Voronoi region and that can see the same feature. To approximate this without actually calculating the Voronoi Diagram, we introduce our first sector merging rule.

Merge rule #1. Merge adjacent sectors which can see the same closest feature.

This results in the smallest number of merges in a practical sense, which emphasizes sector property #3 but not sector properties #1 and #2. If sector property #3 is not too restrictive, then we can further reduce the number of sectors and maximize sector property #2 with the following rule.

Merge rule #2. Merge adjacent sectors which share some common visible feature.

Rule #2 produces the opposite effect as rule #1. The original sectors before merging are shown in Figure 3(a), and the results of applying the rules are illustrated in Figure 3(b) and

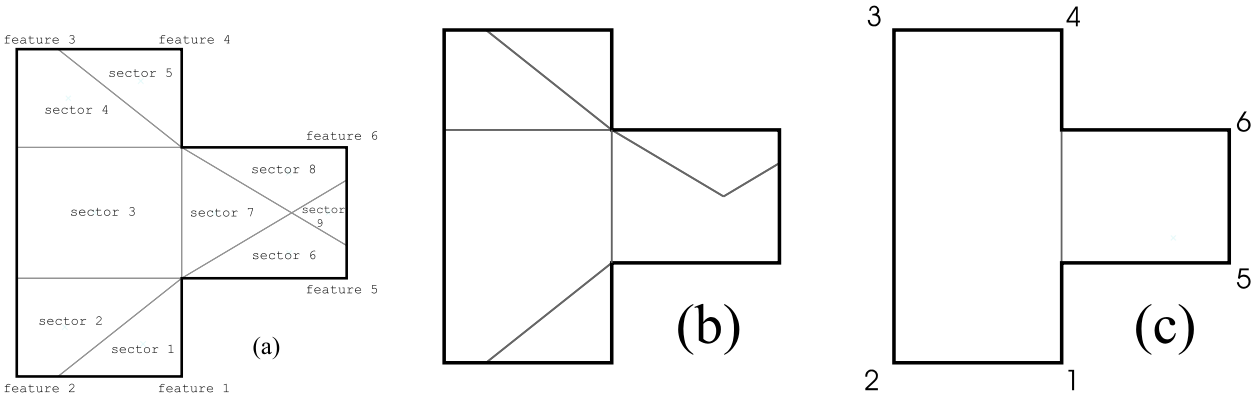

Figure 3: Visibility Sectors before Merging(a), after Merging with Rule #1(b) and Rule#2(c)

For a relaxed visibility sector, we may keep a different number of features after preprocessing. For example, the left sector in Figure 3(c) has four features always available. However, in some parts of the sector, feature #5 and #6 may be also used. 3(c). When rule #1 is applied, the number of sectors is reduced to six, which is the same as the number of features in this case. If we apply rule #2, the resulting number of sectors is two: we cannot have less than two sectors because the environment is not convex.

To maximize sector property #2, it is desirable to keep all the features. Since our modified localizer works best with all the features, we choose to store the union of the features for the original sectors.

Another case with a triangular obstacle inside an environment is shown in Figure 4 where 4(a) is the original, 4(b) results from merge rule #1, and 4(c) results from rule #2. Notice how merge rule #1 simulates the use of the Voronoi Diagram. The total number of sectors is 65, 7 and 3, respectively. Note that after merging with rule #1, the number of sectors is not the same as the number of features.

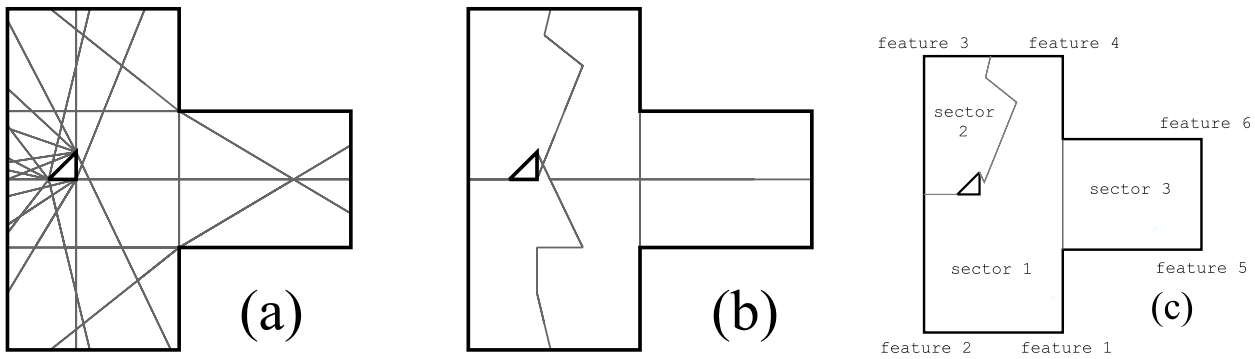

Figure 4: Visibility Sectors before(a) and after Merging with Merge Rule #1(b) and #2(c)

2.3 Modified Localization Algorithm

The localizer requires the following information: a list of sectors where the robot is expected to be and characteristic points extracted from the real scan data. The set of the expected positions

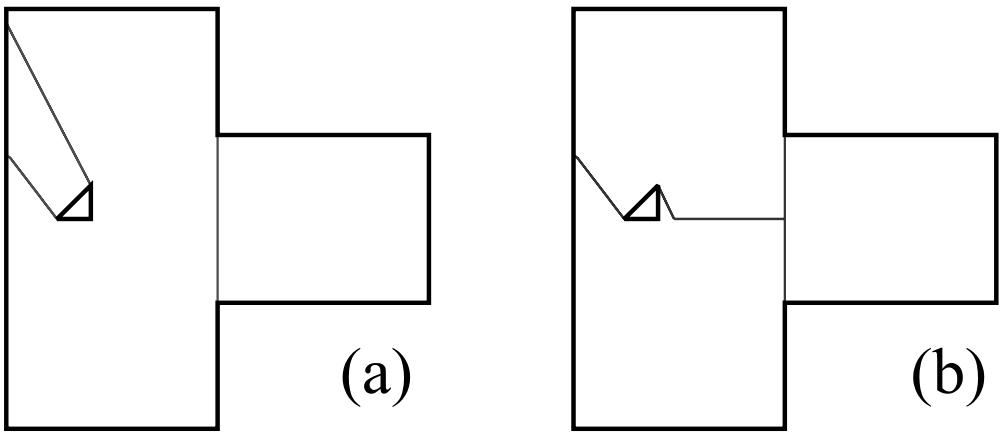

Figure 5: Merging Result with No Rule, Case 1(a) and Case 2(b)

of the robot as it moves is represented by an *uncertainty ellipse* that grows linearly as the robot translates or rotates. In our previous work, localization was performed in two steps: first localize to a sector and then to a configuration. However, we do not need to know which sector contains the robot (i.e., do not localize to a sector) since we can simply try all features visible from those sectors and select the result which is validated by the scanned range data. For example, if the robot is expected to be located in sector 1 or 2 in Figure 3(a), then we know that we can try features 1, 2, 3, 4 and 6. In this case, without determining which sector contains the robot, we try localizing using all the features.

This process requires a *confirm* step to select the correct localization result which will determine if the scanned range data could have been collected at that point. In the first confirm substep, we do a rough test using the distance between the result and the boundary of the uncertainty ellipse, checking if the distance is not greater than a threshold value which allows for some error from sensor noise. In the next step, we compare the actual scan with a *synthetic scan* (a software scan of the model environment from the configuration determined by the localization). The configuration which is the best match will be selected. Note that the distance check comes first because calculating the synthetic scan takes longer than checking the distance, and if robot is not in free space, we cannot compute a synthetic scan.

The pseudo code of our modified localization algorithm based on the relaxed visibility sectors is described in Figure 6. To test the robustness in a partially known/dynamic environment, each of the environments shown in Figure 7 was used for scanning in the localization step while the environment in Figure 3(a) was used during preprocessing. All tests were successful.

The price paid for the flexibility is that the localization may take more iterations than before. The most time consuming step is generating and comparing the synthetic scans. To reduce the number of iterations, we can test the most likely feature pair first. One approach is to first try the sector that contains the subgoal. Or, we can use a string edit distance function to compare the characteristic points in the real scan data with the characteristic points from the preprocessed environment. Next, we can try the closest features first since close features are less affected by sensor noise.

3 Optimizing Multiple Subgoals

Before discussing subgoal optimization, we list the difficulties in localization for multiple robots using sonar range sensors.

```
LOCALIZER
1.  S ← relaxed sector(s) intersecting the
        uncertainty ellipse
2.  F1 ← all features of S
3.  F2 ← all features of real scan
4.  for each pair of (f1 ∈ F1, f2 ∈ F2) {
5.      compute robot configuration
6.      D ← distance from result to ellipse
7.      if (D < distance threshold)
8.          then synthetic scan and compute error
9.  }
10. choose configuration giving smallest error
```

Figure 6: Pseudo Code for Localization

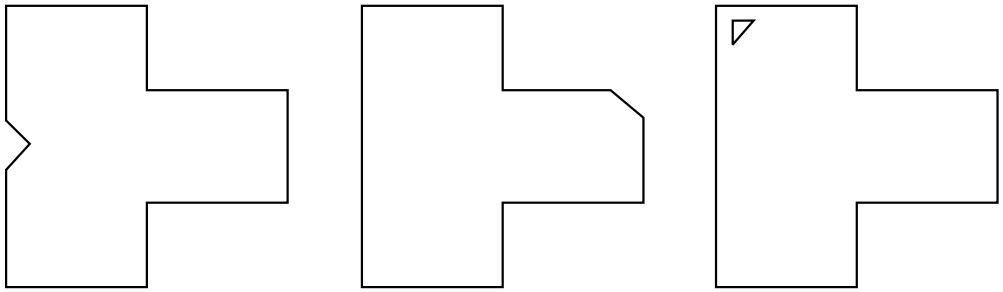

Figure 7: Test Cases of Unknown Obstacle

1. Measurements from the sonar sensors are faulty if the angle between the signal and the wall is not close to perpendicular. This depends on the material of wall.

2. For a 360^o scan, the number of measurements cannot be arbitrarily large. Using fewer scans is critical in localizing faster.

3. To compute the line containing a wall (necessary for our low level localization algorithm), we need at least two scans in theory. In reality, we often need more.

4. Constraints, if any, among multiple robots must be maintained while rearranging subgoals.

5. The robot cannot be closer to the wall than the sensor's minimum range limit.

6. The robot cannot be farther from the wall than the sensor's maximum range limit.

For the first three conditions, we select the subgoal which optimizes the scan angle for all robots (Section 3.1), and for the fourth condition we use subgoal rearrangement (Section 3.2). For the fifth condition, we use a roadmap built on the medial axis to provide the largest clearance in general. The sixth condition is also considered as the second criterion in Section 3.1.

In fact, we cannot avoid reading 'too far' from the sensors unless the environment is small enough. To deal with this, we introduce a new characteristic point `f`. Even with unexpected `f`'s included in the characteristic points, we can localize correctly as long as a nearby feature is correctly scanned. This shows that our localizer is robust with respect to sensor range limits as well as to unknown obstacles.

3.1 Optimizing Scan Angle

This is better explained with single robot case. The original algorithm simply selects the subgoal closest to the goal whose uncertainty ellipse is not in collision. In other words, if the uncertainty ellipse grows large enough that it intersects an obstacle, then the path node right before the collision is chosen as the subgoal. This is based on the simple assumption that the robot can scan at least one feature anywhere in the environment, which is not true in reality due to the mentioned difficulties #2, #3 and #6.

To solve these problems, we score each node on a path by checking how well the robot can scan a feature which satisfies the range constraints. For example, if a feature can be scanned from a node whose line of sight to the feature exactly bisects the space between the walls adjacent to the feature, then it is considered a good node for localization. After scoring each node, the best one is selected as the subgoal.

For a feature to be available, it needs to satisfy two conditions. One is that the distance from the node to a feature needs to be within the sensor range limit (sensor range condition). The second is that the walls adjacent to a feature should be longer than a threshold value to enable at least two readings along that wall (wall length condition). The nodes are scored as follows. If a path P is composed of n nodes $(N_1, N_2, \ldots, N_n)$ and for the ith node N_i, if m features $(F_{i1}, F_{i2}, \ldots, F_{im})$ are available with adjacent walls W_{i1}, W_{i2}, then the bias B from the optimal case is defined as follows (see Figure 8 inset)

$$\begin{gathered} \theta_1 = \angle\overline{N_i F_{ij}} - \angle W_{i1} \\ \theta_2 = \angle\overline{N_i F_{ij}} - \angle W_{i2} \\ B = \mid \theta_1 - \theta_2 \mid \\ \text{where } \; i = 1 \ldots n, \;\; j = 1 \ldots i_m \end{gathered} \tag{1}$$

The scan angle subgoal optimization searches for a node i such that B is minimum for any j and which satisfies the sensor range condition and the wall length condition. Figure 8 is a screen shot of a simulation where the subgoal (center of the two robots) is obtained without this optimization. After scan angle subgoal optimization, the subgoal is moved as shown in Figure 9(a). Before optimizing, $\theta_1 = 15^o$, $\theta_2 = 75^o$ and $B = 75^o - 15^o = 60^o$. On the other hand, in Figure 9, the bias angle is close to zero.

3.2 Rearranging Multiple Subgoals

If two robots are in the same relaxed sector and one robot is blocking the other robot from scanning a selected feature, then either one of the robots should localize using a different feature, or the robots should rearrange their relative positions so they can both scan the same feature. In the former case, since the availability of another feature is not guaranteed, one robot may randomly move until it leaves the sector. This is not considered in this paper since it is undesirable to move the robot without localizing first. We designed the relaxed sector to provide the most effective feature in the general situation. So, our proposed solution is to share the feature by rearranging subgoals when the robots are inside the same sector.

For a robot R_i, let S_i be the space allowed to legally maneuver and let A_i be the angle range for scanning a feature without being blocked by the other robot. Note that the angle space is important in scanning at least two points on each wall adjacent to a feature. For all robots, our goal is to optimize the subgoals N_i so that

$$\begin{gathered} \max \;\; (\min \{A_1, A_2, \ldots, A_n\}) \\ \text{for } \; N_i \in S_i \;\; \text{and} \;\; i = 1, \ldots, n \end{gathered}$$

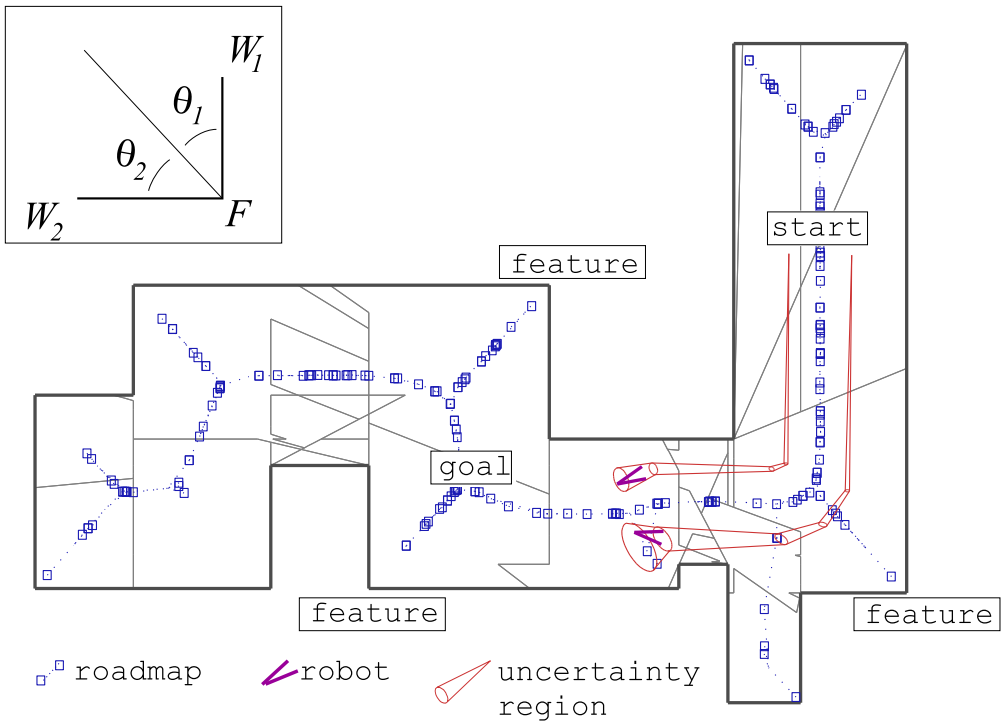

Figure 8: Two Robot Case, before Optimization

The implementation of the optimization depends on the constraints that must be maintained among robots. For example, if the distance between robots must be constant, then we can rotate the robots around the center of the two robots. If the robots must remain on the same X coordinate, then we can move the robots in the Y direction. In general, they can simply move along their paths.

Before the subgoal rearrangement shown in Figure 9(a), the sensors' effective ranges are not long enough to scan the ends of the hallway. This forces them to localize to the bottom right corner and the left robot cannot get enough scans to localize using the feature. The result of the rearrangement is shown in Figure 9(b) which is obtained by the algorithm described in Section 4. If the two robots are in different sectors, then they will use different features. As shown in Figure 10, no subgoal rearrangement is necessary since the robots are in different sectors and use features A and B.

4 Implementation and Experiments

Figures 8, 9 and 10 show the environment used for the hardware experiments as well as the simulations. For software development, we use C++ in linux with the LEDA library [13] for geometric calculations and SAPHIRA [17] for robot communication. For the hardware experiments, two AmigoBots [3] are used in a partly real hallway and a partly experimental environment. These two robots communicate with two separate computers through wireless modems because SAPHIRA supports connection to only one robot. The two computers are connected to a computer which performs all calculations over the network. For range measurements, sonar sensors built into the AmigoBot were used. The size of the environment is 11.3 by 7.62 meters (444 by 300 inches), and the length of the robot is approximately 33 cm (13 inches).

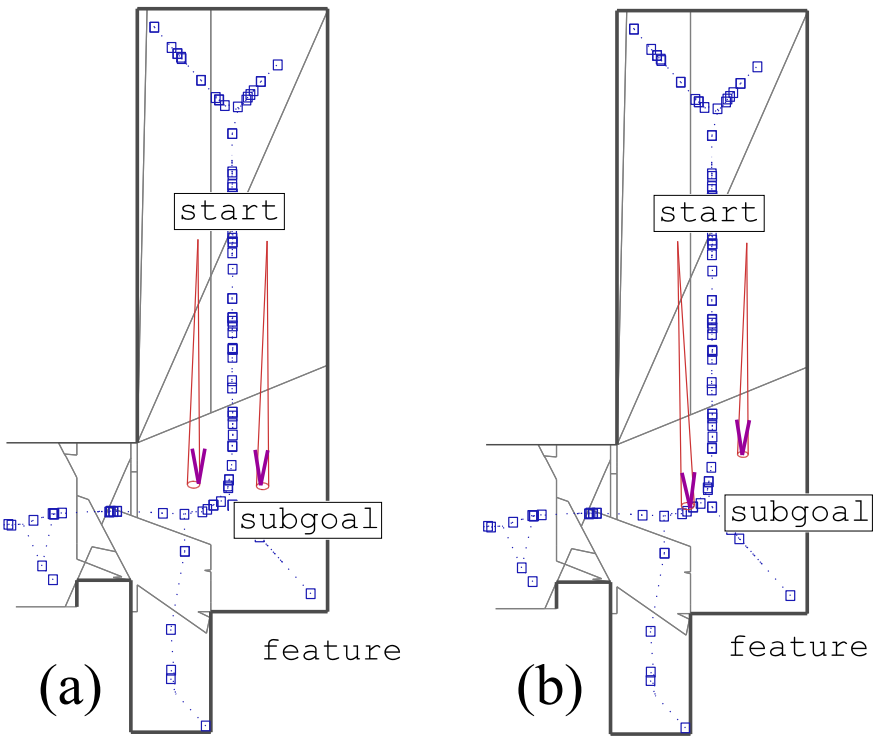

Figure 9: Two Robot Case, after Optimizing Scan Angle(a) and Rearrangement(b)

Roadmap Based on MAPRM As briefly described in Section 2, since we are interested in navigation where the distances between the robots remain almost fixed, we generate paths for multiple robots from the path for a single robot.

The automatically generated roadmap provided by the OBPRM motion planner package [2] (using medial axis node generation) is imported during the preprocessing step. This enables the program to find a path in a very complex environment. The resulting roadmap contains many closely–spaced nodes, which is useful in choosing an optimal subgoal.

Uncertainty Regions Given a path, uncertainty ellipses at each node are first calculated. The *uncertainty region* is the union of the ellipses sliding along the path which grows linearly. Since we cannot compute the union of an infinite number of ellipses, we just connect the ellipses at each node by line segments tangent to the ellipse. The uncertainty region is used to enforce the constraints among the robots. First, the path for a single robot is multiplied for multiple robots and the uncertainty region for each path is computed. Then, we can detect where constraints are violated and select the subgoals for multiple robots. These subgoals will be rearranged before localization.

In our experiments, we assumed that the distance between two robots can be close as long as they are not in collision, and the distance can be reasonably large which allows for a simple implementation of the subgoal optimizations. This corresponds to the situation where each of the robots has an arm with several links and joints, and the robots are carrying an object.

Optimizing Subgoals Implementation of the scan angle optimization is straightforward and involves calculating B in Equation (1) for each node and choosing the minimum value for each node. This approach uses a bisector perpendicular to the line segment connecting the center of the robots and the feature. If the path of a robot crosses this bisector, then the subgoal is moved to the intersection. For the other robot, a point on the bisector close to the subgoal before rearrangement

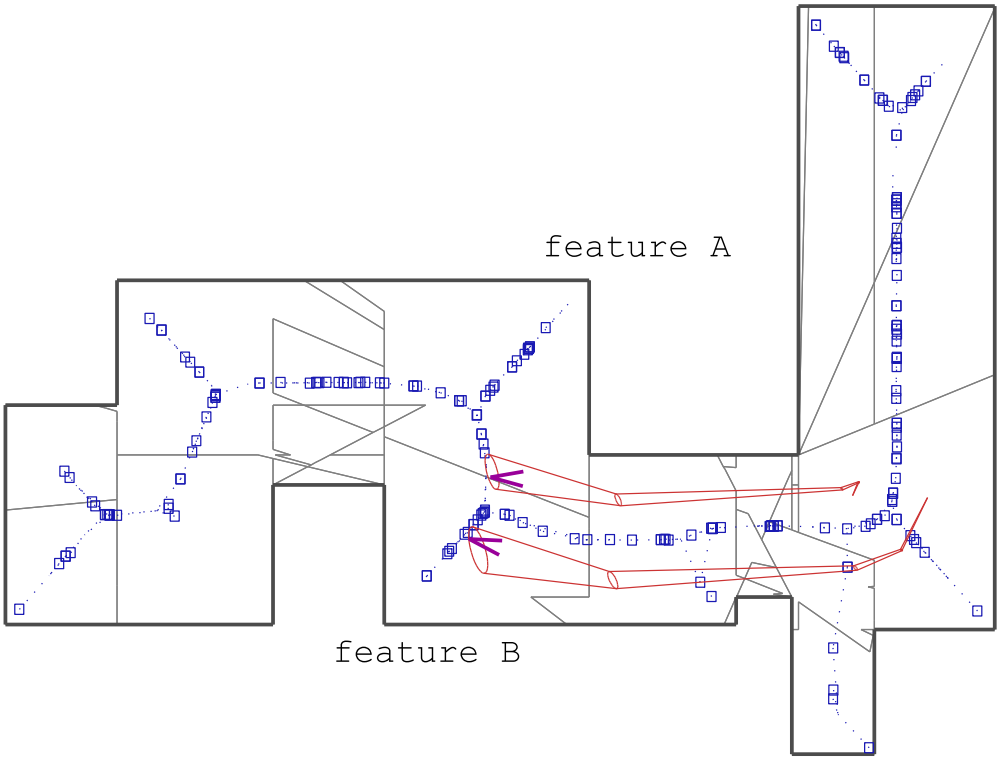

Figure 10: Subgoals for Second Localization

is selected.

Filtering Sonar Sensor Measurements To obtain range measurements, we used two sonar sensors with opposite orientation provided on the body of the AmigoBot. Each firing of a sensor corresponds to a line segment shown in Figure 2, and the robots were rotated at the fixed position to cover 360^o range. Our hardware experiment used 72 measurements which means the angle step of rotation is 5^o; only 36 turns were actually necessary since we used two sensors.

Among the acquired range measurements, only a small number of data points were actually useful. The sonar sensor signal tends to be very noisy if the wall is not perpendicular to the scan or if a corner is present nearby. However, our localizer works if two walls adjacent to a feature are identified. This allowed us to design a very simple filter which in fact performs the least amount of filtering required.

The pseudo code for our implementation is shown in Figure 11. This is based on the assumption that the two subgoal optimization steps will always enable scanning two close walls for each feature so that we can identify them by starting from the closest scan data. The number of repetitions depends on the number of robots which may appear to each other as additional close walls. Another property of the sonar sensor used here is that faulty data is usually farther than the actual range. This is explained by considering the multipath effect on a slanted wall. Currently, we do not use any mathematical method such as least squares or principal analysis, which could be employed as necessary.

4.1 Results

For the environment and the subgoals shown in Figure 9(b), localization was done within the allowable error. Figure 12(a) and 12(b) illustrate the actual sonar range sensor data in the robot's

```
FILTER
1.  for 3 ~ 5 times (depending on situation) {
2.      MS ← unprocessed scan, minimum range
3.      check a few scans around MS
4.      if they form line segment (L) {
5.          repeat {
6.              LS ← left scan adjacent to MS
7.              if LS is farther than MS {
8.                  move LS to be on the line L
9.                  MS ← LS
10.             }
11.         } until no farther LS found
12.         do the same for right direction
13.     } (end if)
14. } (end for)
```

Figure 11: Pseudo Code for Filtering Sensor Data

coordinate system, before and after filtering. The environment is shown in Figure 9, and the robot is located nearby the right side subgoal. In the figure, the longest segments denote an unreturned signal, which is 2.54 meters (100 inches). The resulting error is 13 cm (5 inches) north for the right robot and the same amount south east for the left robot. Using mathematical techniques such as least squares will help reduce this error.

5 Conclusion

In this paper, we described the design and implementation of a method for navigation and localization for multiple robots in a partially known indoor environment, such as a home or office. Our method requires only inexpensive sonar range sensors. Our framework is based on our previous work, and two new steps for subgoal optimization are introduced to enable localization of multiple robots. Our simulation and hardware experiments show the practicality and potential of our approach in the presence of sonar sensor noise.

Our current research includes using local minimum characteristic scan points as additional features for localization, and optimizing the scan range to obtain only the required number of scans around desired feature(s). We also have more extensive hardware experiments planned.

References

[1] F. Ababsa and N. Bouguechal. Environment modeling and localization technique for an autonomous mobile robot. In *IEEE Internation Conference on Control Applications*, pages 1338–1342, 1998.

[2] Nancy M. Amato. motion planning group webpage. `http://www.cs.tamu.edu/faculty/amato/dsmft`.

[3] Amigobot. http://www.amigobot.com/.

[4] E.G. Araujo and R.A. Grupen. Feature detection and identification using a sonar-array. In *ICRA*, pages 1584–1589, 1998.

[5] E.G. Araujo and R.A. Grupen. Feature extraction for autonomous navigation using an active sonar head. In *ICRA*, pages 3823–3828, 2000.

[6] M. Bennewitz, W. Burgard, and S. Thrun. Optimizing schedules for prioritized path planning of multi-robot systems. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 271–276, 2001.

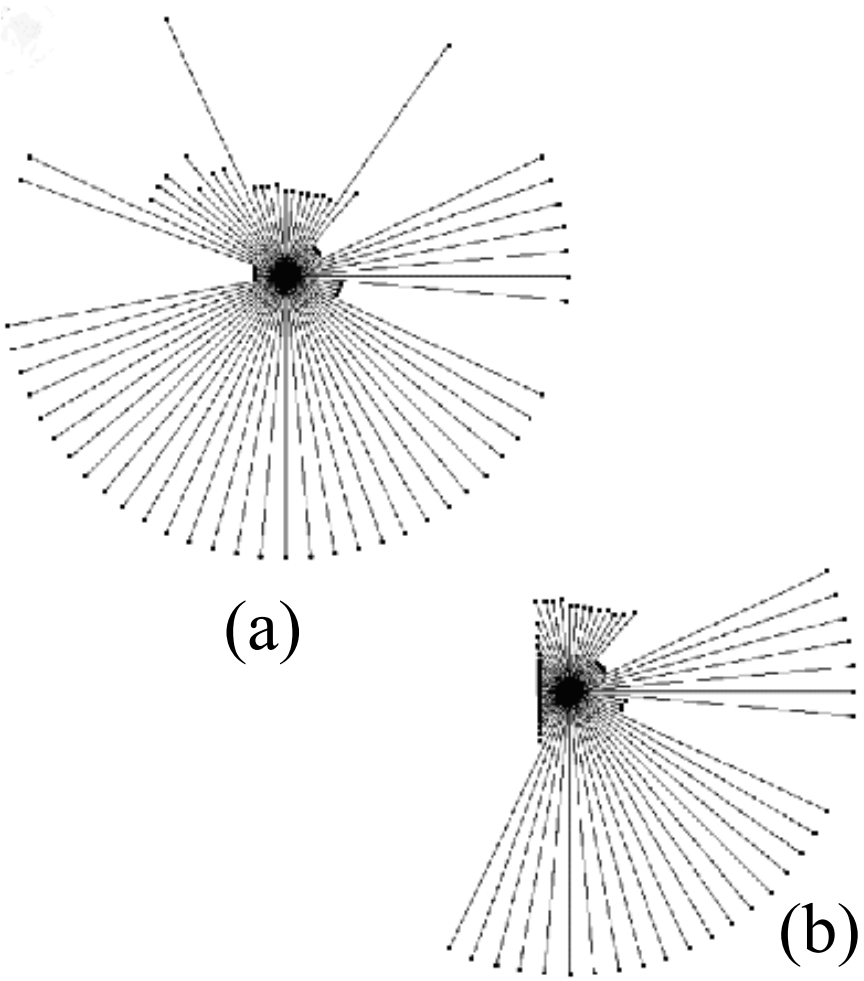

Figure 12: Sonar Data (a) before and (b) after filtering.

[7] L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. Technical Report TR 00-003, Department of Computer Science, Texas A&M University, 2000.

[8] P. Jensfelt, D. J. Austin, O. Wijk, and M. Andersson. Feature based condensation for mobile robot localization. In *ICRA*, 2000.

[9] J. Jongusuk and T. Mita. Tracking control of multiple mobile robots: a case study of inter-robot collision-free problem. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2885–2890, 2001.

[10] J. Kim, N. M. Amato, and S. Lee. An integrated mobile robot path (re)planner and localizer for personal robots. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3789–3794, 2001.

[11] S.M. LaValle, J.H. Yakey, and L.E. Kavraki. A probabilistic roadmap approach for systems with closed kinematic chains. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1999.

[12] M. Shimada M. Yamamoto and A. Mohri. On-line navigation of mobile robot under the existence of dynamically moving multiple obstacles. In *International Symposium on Assembly and Task Planning*, pages 13–18, 2001.

[13] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, New York, 1998.

[14] T. Nakamura, A. Ebina, T. Ogasawara M. Imai, and H. Ishiguro. Real-time estimating spatial configuration between multiple robots by triangle and enumeration constraints. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, volume 3, pages 2048–2054, 2000.

[15] T. Nakamura, S. Takamura, and M. Asada. Behavior-based map representation for a sonar-based mobile robot by statistical methods. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 276–283, 1996.

[16] Igor E. Paromtchik and Hajime Asama. Optical guidance system for multiple mobile robots. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 3, pages 2935–2940, 2001.

[17] Saphira. http://www.ai.sri.com/~konolige/saphira/.

[18] P. Švestka and M. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1995.

[19] Z. Yi, H.Y. Khing, C.C. Seng, and Z.X. Wei. Multi-ultrasonic sensor fusion for multiple robots. In *IEEE Intelligent Vehicles Symposium*, pages 387–391, 2000.