Robust Geometric-Based Localization in Indoor Environments using Sonar Range Sensors*

Jinsuck Kim   Roger A. Pearce   Nancy M. Amato

Department of Computer Science
Texas A&M University
College Station, TX 77843–3112
{jinsuckk,rap2317,amato}@cs.tamu.edu

Abstract

In this paper, we describe a method for navigation and localization of a mobile robot using sonar sensors in an indoor environment. This is an enhanced version of our previous method which assumed a perfectly known environment and perfect sensor data. We remove these assumptions by computing a roadmap and selecting geometric features of the environment for localization that are robust in terms of known sensor limitations and uncertainty. In particular, our roadmap-based navigator and localizer have been redesigned to work cooperatively. To identify geometric features, a simple sensor data filter is designed. We present simulation and hardware experiments for a robot equipped with inexpensive sonar sensors in a real environment.

1 Introduction

In the near future, more manufacturing tasks will become automated as technology improves. Independent service robots will prove useful in diverse ways such as search and rescue, cleaning, and assisting the elderly or those with physical disabilities. These tasks are examples of potential uses of low cost mobile robots in indoor environments such as the home or office.

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. Popular low cost sensors are sonar range sensor(s). Though widely used, these sensors have limitations such as a restricted range and scan angle incidence.

There are two approaches related to our localization work using sonar sensors. One is based on the probabilistic model of motion and perception, assuming the environment is Markov (past and future data are conditionally independent if one knows the current state). A well-known implementation is Monte Carlo localization [6] which is based on particle filters. One feature of this approach is that global localization of the robot is possible even if the initial position of the robot is not known. Several variants have been proposed for the sonar sensor case, including efficient tracking of landmarks using additional Kalman filters [12] and a feature based condensation method [7]. The combination of statistics-based methods and the sonar sensor model is said to result in more effective localization in terms of computation time and accuracy.

Another related approach to localization focuses on the nonlinear behavior of sonar range sensors and extracts geometric features of the environment using a filter based on the actual sonar sensor model. Techniques include principal component analysis [11], Dempster's rule of combination [14], model matching by coordinate transformation [1], and applying a Kalman filter for feature identification [5]. In fact, some of these techniques can be regarded as a perceptual model of Monte Carlo localization.

Our approach is Markov and utilizes actual sonar sensor modeling, but it differs from Monte Carlo localization in that we apply geometric techniques for both the motion model and the perceptual model. Because it is not based on statistics, our localizer requires fewer measurements and less computation. Another advantage is that our localizer is combined with a path planner in such a way that the two components assist each other: we plan a path so that the robot moves through a localizable area (the navigator aids the localizer), and a collision–free path from the start to the goal is obtained by localizing when necessary (the localizer aids the navigator).

Our framework is an extension of our previous work described in [8, 9] which describes navigation and

*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.

localization for one robot in a perfectly known environment with ideal range sensors. We assume that a map of the environment is given and the initial position and orientation of the robot is known. Our preprocessor assumes that the environment can be represented by simple polygons. A path is extracted from a pre–computed roadmap, and geometric feature information about the environment is pre–processed and stored for efficient localization. Unlike other approaches for handling sonar sensor limitations, we use a very simple filter that is specialized for our localizer. The proposed system has been implemented in a real environment using an AmigoBot robot.

2 Feature Based Localization

The pseudo code in Figure 1 describes our system at a high level. The overall strategy is the same as in our previous work [8, 9], the only difference being the sophistication of the preprocessing and the subgoal determination steps. In a preprocessing phase, a roadmap is generated and the environment is subdivided into sectors which contain useful information for localization. Initially, the robot is in its starting configuration and the goal configuration is known. In the main loop, a collision–free path is extracted from the roadmap. If necessary, a subgoal is selected. The subgoal is determined so that the robot is expected to be able to move there safely, localize there, and then continue moving toward the goal.

```
NAVIGATOR(start, goal)
preprocess environment
while goal is not reached {
    extract path from start to goal
    determine localizable subgoal in the path
    drive robot to subgoal and stop
    scan and localize
    set start to current configuration
}
```

Figure 1: *Pseudo-code for our system.*

2.1 Definitions

In this section, a perfect range sensor (such as laser) is assumed; this assumption will be removed in Section 3. The basic building blocks of our localizer are *characteristic points* which are obtained from 360° range scans of the environment. Local minimum `m` (maximum `M`) points are scans where both adjacent scans are farther (nearer). To determine if the ranges of two neighboring scans are discontinuous, we use a threshold value for the range check (the pair `Dc(cD)` indicating an increase(decrease) in range). Figure 2

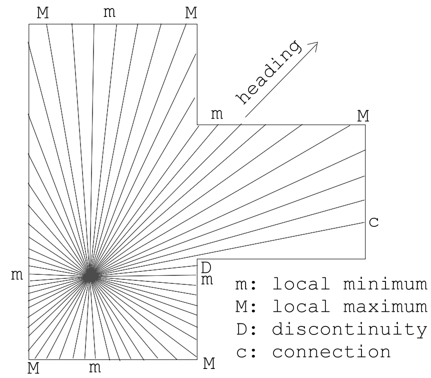

Figure 2: *Simulated sensor readings and corresponding characteristic points* `mMmMmMmMmDcM`.

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 it provides two–dimensional information when used with adjacent characteristic points [9].

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 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 does not 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 (original) visibility sector definition is unique, the relaxed sector subdivision is not unique. That is, if two adjacent sectors can see a common feature, we may or may not merge them. If they are merged, more features are available and localization is more robust. There are always trade-offs, however. To investigate these issues, 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, and increases robustness in the presence of 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.

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. It is not difficult to see that random merging can lead to very diverse results. As an example, consider the two cases shown in Figure 5 in which the size and shape of the relaxed sectors is very different although both were constructed from the visibility sectors shown in Figure 4(a).

To satisfy sector property #3, we need to keep the distance from a feature to a sector boundary as small as possible. One solution is merging sectors that are in the same Voronoi region (features are Voronoi sites) since they share the same closest feature. To approximate this without actually calculating the Voronoi regions, 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 and strictly satisfies sector property #3. In many cases, a sector contains several features which allows further merges and maximizes sector property #2.

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

Rule #2 is the counterpart of 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 (c). The sensor range limit is larger than the length of the diagonal of the environment.

For each relaxed visibility sector, we may keep a different number of features after preprocessing. For example, the left sector in Figure 3(c) has four fea-

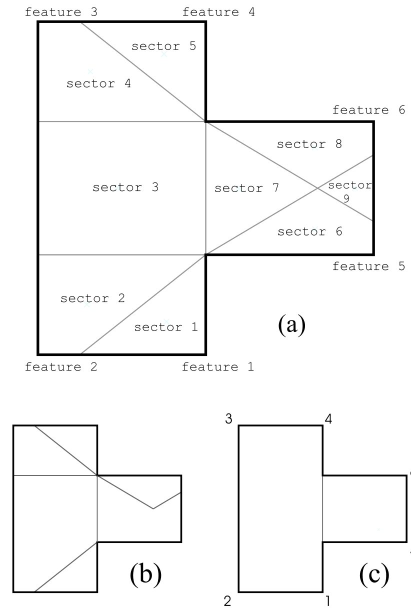

Figure 3: *Visibility Sectors (a) before Merging, (b) after Merging with Rule #1 and (c) Rule #2.*

tures always available. However, in some parts of the sector, feature #5 and #6 may be also used. 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 fewer than two sectors because the environment is not convex.

To maximize sector property #2, it is desirable to maximize the number of 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 environment with a triangular obstacle inside an environment is shown in Figure 4 where 4(a) is the original, 4(b) shows the results from merge rule #1, and 4(c) shows the 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. Unlike the previous case, after merging with rule #1, the number of sectors is not the same as the number of features (there is a very small sector to the right of the obstacle).

2.3 Modified Localization Algorithm

Our 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. A region that contains the expected position of the robot 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 we localized to a sector and then to

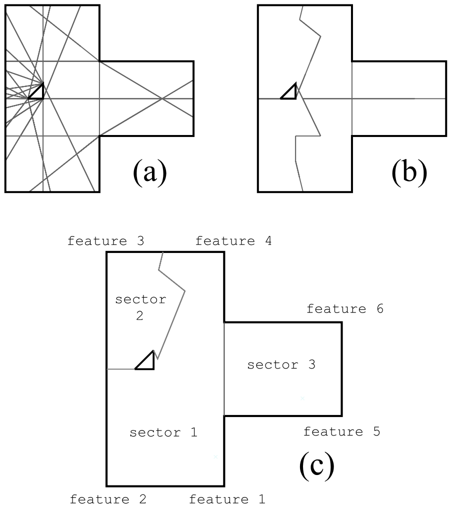

Figure 4: *Visibility Sectors (a) before and after Merging with Merge Rule (b) #1 and (c) #2.*

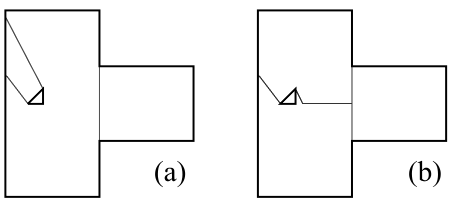

Figure 5: *Merging Result with No Rule, (a) Case 1 and (b) Case 2.*

a configuration. However, we do not need to know which sector contains the robot (i.e., it is not necessary to localize to a sector) since after trying all features visible from those sectors, the correct result can be validated. 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.

After the localization for the possible features, a *confirm* step is used to select the correct result. In the first confirm substep, we perform a coarse filter 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 the error from Gaussian sensor noise. Next, 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 of the best match will be finally selected.

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

```
LOCALIZER
S ← sector(s) intersecting uncertainty ellipse
F1 ← all features of S
F2 ← all features of real scan
for each pair of (f1 ∈ F1, f2 ∈ F2) {
   compute robot configuration for (f1, f2)
   D ← distance from result to ellipse
   if (D < distance threshold)
      then compute error from synthetic scan
}
choose configuration giving smallest error
```

Figure 6: *Pseudo Code for Localization.*

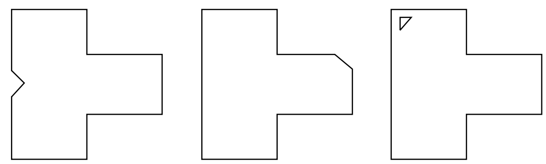

Figure 7: *Test Cases for Unknown Obstacles.*

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.

3 Finding Localizable Subgoals

In this section, we remove the assumption that range sensors are perfect. Two major problems of sonar sensors are limited range (minimum and maximum) and incidence angle (works only if incidence angle is close to 90 degrees). An example sonar sensor scan from the AmigoBot is shown in Figure 9(a) with the robot placed at the subgoal of Figure 12(b).

3.1 Feature-based Roadmaps

The paths our robot will follow are extracted from a roadmap (a graph). We need to compute a roadmap which allows the robot to localize when necessary by not allowing the path to stray far from the features. This requires that a certain percentage of the roadmap nodes should be near the wall, where the clearance depends on the sensor's minimum range and the size of robot.

Since our path planner provides several options for generating roadmap nodes, we choose to use OBPRM [2] (most nodes are within a fixed distance of the obstacle) and MAPRM (nodes are on the medial axis of the free space) [3]. Figure 8(a) shows the combination of the medial axis and obstacle-based roadmap. Approximate localizable areas have been estimated based on the sensor's maximum range and are illustrated by hatched areas in Figure 8(b). This environment will be used for simulation and experiment.

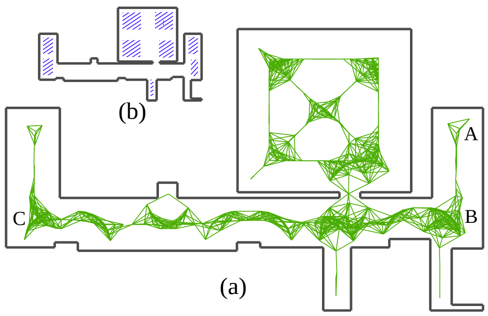

Figure 8: *(a) Roadmap and (b) Approximately Estimated Localizable Areas.*

3.2 Refinement Subgoal

For the incidence angle condition, the roadmap needs to provide nodes for localization which allow the robot to scan at least two local minimum points (incidence angle is almost 90°) on walls – these be used to calculate the global position of the local maximum point by filtering.

In Figure 9(a), 60 scans were taken with an angular step of 6°. Our sensor works if the incidence angle is larger than approximately 85° since the number of correct scans around a local minimum is 3∼4. Our filter utilizes a small number of correct scans to extrapolate the position of the wall, and is based on the assumption that the subgoal satisfies the following condition illustrated in Figure 11. Here the subgoal is projected to the points m_1, m_2 and the end points of the walls are w_1, w_2. Thus, for localization, we require

$$\overline{fm_1} < \overline{fw_1} \quad \text{and} \quad \overline{fm_2} < \overline{fw_2},$$

which means that the robot should be able to scan local minimum points on the walls adjacent to a feature. The pseudo code for our filter is shown in Figure 10. Currently, we do not use any method such as least squares or principal component analysis, which could be employed as necessary.

A subgoal which only guarantees collision avoidance (as in our previous method) is shown in Figure 12(a), and the new subgoal selected by the incidence angle condition is shown in Figure 12(b). Note that the sensor range limit condition is satisfied implicitly: if a sector contains no nearby features, as in Sector 3 in Figure 12(b), then no subgoal refinement will be attempted.

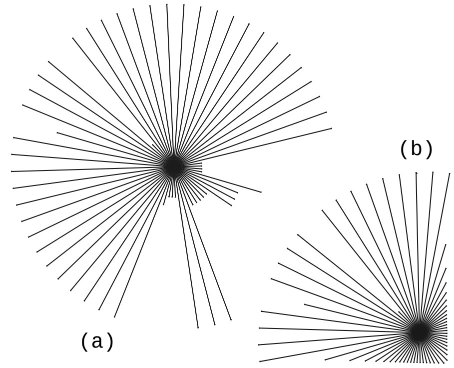

Figure 9: *Sonar sensor measurements (a) before and (b) after filtering.*

```
FILTER
for 3 ∼ 5 times {
    MS ← unprocessed minimum range scan
    if a few scans around MS form a line (L) {
        repeat {
            LS ← left scan adjacent to MS
            if LS is farther than MS {
                move LS toward sensor to be on L
                MS ← LS
            }
        } until no farther LS found
        do the same for right direction
    } (end if)
} (end for)
```

Figure 10: *Pseudo Code for Filtering Sensor Data.*

4 Implementation and Experiments

For software development, we used C++ in linux with the LEDA library [10] for geometric calculations and SAPHIRA [13] for hardware communication. For the hardware experiments, an AmigoBot [4] was used in the hallway in our building (shown in Figure 12). For range measurements, sonar sensors provided by the AmigoBot were used. The size of the environment shown in Figure 8 is 21 by 13 meters and the maximum scan range of the sonar sensor was set to 2.54 meters.

In the hardware experiment, the robot moved from A to C in Figure 8(a) and localized at B with an acceptable error. In Figure 9(b), the extrapolated line segments are not exactly perpendicular to each

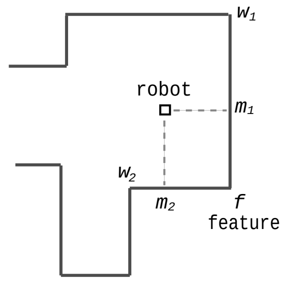

Figure 11: *Adjacent Wall Length Condition.*

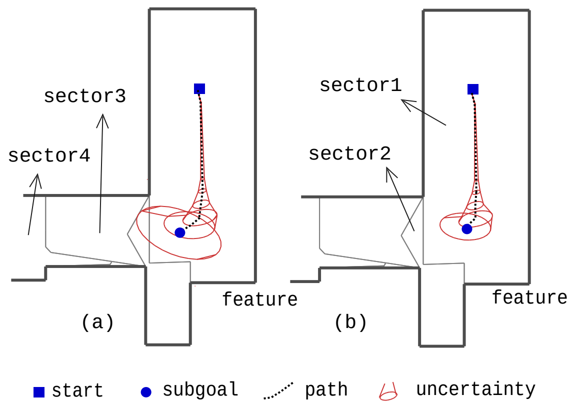

Figure 12: *Relocation of Subgoal Using Wall-Length Condition, (a) before and (b) after.*

other due to some noise in the measurements. However, the resulting error was 10cm north from the actual robot position, and the localization was successful. The robot moved through the hallway (from B to C) without localization by using a following collision–avoidance maneuver.

The method is efficient. In the examples studied here, the visibility sectors are computed in 10 seconds, and the sectors are merged in about 30 seconds. The roadmap contains 299 nodes and 2246 edges, and its construction takes about 10 seconds including node generation and connection in a Pentium III machine.

5 Conclusion

In this paper, we described the design and implementation of a method for navigation and localization for a robot in a partially known indoor environment, such as a home or office. Our simulation and hardware experiments show the practicality and potential of our approach in the presence of sonar sensor limitations.

Our current research includes using local minimum characteristic points as additional features for localization so that the localizable area in Figure 8(b) is significantly increased, and optimizing the range of scan angle so that we only perform the minimum required number of scans around the 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] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 155–168, 1998.

[3] Nancy M. Amato. Motion planning group webpage. `http://www.cs.tamu.edu/faculty/amato/`.

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

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

[6] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1322–1328, 1999.

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

[8] 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.

[9] S. Lee, N. M. Amato, and J. P. Fellers. Localization based on visibility sectors using range sensors. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3505–3511, 2000.

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

[11] 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.

[12] S. Roumeliotis and G. Bekey. Bayesian estimation and kalman filtering: A unified framework for mobile robot localization. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2985–2992, 2000.

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

[14] 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.