Mohamed Dekhil
Department of Computer Science
University of Utah
Salt Lake City, Utah 84112, USA

Tarek M. Sobh
Department of Computer Science and Engineering
University of Bridgeport
Bridgeport, CT 06601, USA

ABSTRACT

Tolerance analysis constitutes an essential task in designing and building robust sensor systems for robotic applications. In most current practices, tolerance analysis is considered after the system is designed and implemented. This leads to poor system integrity and lower overall reliability. In this paper we discuss several aspects of tolerance analysis for sonar sensors used for mobile robot applications. We show how to embed tolerance analysis within a distributed control scheme. Simulation results of applying this model are presented with a brief discussion and conclusion on these results.

INTRODUCTION

In any closed-loop control system, sensors are used to provide the feedback information that represents the current status of the system and the environmental uncertainties. The main components in such systems are the transformation of sensor outputs to the decision space and the computation of the error signals and the joint-level commands. For example, the sensor readings might be the current tool position, the error signal is the difference between the desired and current position at this moment, and the joint-level command is the required actuator torque/force.

In previous work we proposed a distributed control scheme for mobile robots[1,2]. In this control scheme, several controllers (clients) are working in parallel, competing for the server. The server selects the command to be executed based on a dynamically configured priority scheme. Each of these clients has a certain task, and can use the sensor readings to achieve its goal. A special client with the task of avoiding obstacles is assigned the highest priority.

including time and accuracy requirements can be issued to such processes and the built-in analysis will be used to satisfy these requests (if feasible). Thus, tolerance analysis is an integrated component of the model which help generate more robust and reliable systems.

Tolerance analysis has been addressed by many researchers. Brooks and Iyengar proposed an averaging algorithm for multidimensional redundant sensor arrays[3]. This algorithm provides fault tolerance sensor integration. Prasad et. al. proposed a sensor model based on functional characterization of fault tolerant integration in distributed sensor networks[4]

In this paper, we incorporate tolerance analysis and measures into the sensor model. This provides quantitative measures for the accuracy of the of measured data. It also serves as the basis for devising sensing strategies to enhance the measured data for localization and map construction.

TOLERANCE ANALYSIS FOR SONAR SENSORS

Sonar sensors have been widely used in several robotic applications. This is due to their low cost and reasonable reliability. However, sonar sensors, like most ultrasonic sensors, have several drawbacks. One problem is that the measurements depends on the speed of sound, which vary according to the atmospheric conditions such as temperature and humidity. This usually results in inaccurate and inconsistent readings. Another problem is that the ultrasonic echoes might cause the sensor to measure totally incorrect values.

The use of ultrasonic sensors for mobile robots has been investigated by a lot of researcher[5,6,7]. The main goal is to increase the accuracy and reliability of these sensors, and to filter the noise and echoes to get more consistent data.

The most important criteria in any sensor system are *time* and *accuracy*. Time is the time elapsed between issuing a read request to the logical sensor and the reply to that request. This time depends on the physical aspects of the sensory system, and on the sensing strategy implemented in the logical sensor. Tolerance is defined in this scheme as the region in which the measurement resides.

The following are some variables that will be used in the tolerance analysis for our experiment.

- v_s: speed of sound.
- y_{max}: maximum distance in our indoor environment.
- y_{min}: minimum distance in our indoor environment.
- t_m: the maximum time to get a measurement by the physical sonar sensor.

$$t_m = \frac{2y_{max}}{v_s}$$

- v_r: the linear velocity of the robot in meter/sec.
- Ω_r: the angular velocity of the robot in rad/sec.

In most cases, we cannot satisfy both requirements at the same time. Since the physical sensor has its accuracy limitations, therefore, we might need to get several readings regarding the same measured point to increase the accuracy. This of course will increase the time of measurement. In case of multisensor system, the accuracy can be increased by considering the readings from more than one sensor. In such cases, we should consider the time to fuse the data.

In this analysis, we will ignore measurement noise, and errors due to the interference between the sensors. Also, we will use a simplified beam pattern for the sonar sensors which is a triangle shape centered at the center of the sensor. Our goal here is to be able to determine the location of the measured point within a certain tolerance. Also, we would like to locate edges and door ways within a reasonable tolerance. First, the case of using one sensor will be considered, then the use of multiple sensors to get more accurate measurements will be discussed.

USING ONE SONAR

Figure 1 shows the simplified beam pattern of a sonar sensor. We assume that the sensor will return the distance y of the closest point from the center of the sensor within a tolerance region 2x, where $x = y \tan\theta$.

It is clear that the tolerance area depends on the angle θ and the distance y. However, the angle is fixed for most sonar sensors (It may vary according to the operating frequency.) In our experiment for example $\theta = 11$. Also, there is physical limitations on the minimum distance y_{min} which means there is an upper limit on the accuracy that we can get with the sensor. The upper limit is:

$$x_{min} = y_{min} \tan\theta$$

There are several ways to minimize the tolerance region and to detect the existence of edges within this region. One way is to move in the y direction towards the measured point. Another way is to move small movements in the *x* direction, and a third technique is to rotate with small angle ϕ. In each case, the readings are combined to get smaller tolerance region. Now, let's discuss each of these techniques in more details.

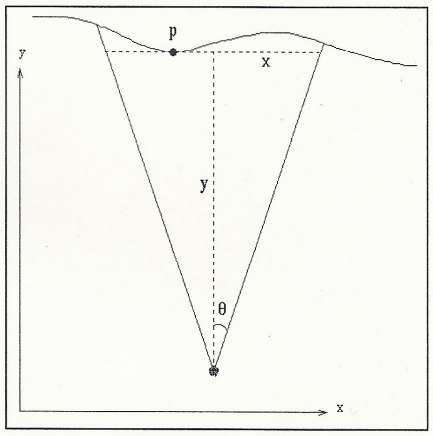

Figure 1: The simplified sonar model

Translation in the y direction

Moving Δy in the y direction towards the measured point, the tolerance region is decreased by:

$$\Delta x = \Delta y \tan\theta$$

This is shown in Figure 2 where y is the initial distance and x is the initial tolerance region, and y' is the new distance and x' is the new tolerance region. The time to make this movement is equal to:

$$t_y = \frac{\Delta y}{v_r}$$

and total time to make this reduction in the tolerance region is equal to

$$t = t_y + t_d$$

If the new distance y' is different than $y-\Delta y$, this means that we encountered an edge or a door way. Figure 3 shows different situations in which this may occur.

In this case, the edge can be located with tolerance $2\Delta x$ since the edge may be at either side as in cases 1 and 3 of Figure 3 or at both side as in case 2 of the same figure. To determine on which side the edge is located, we can move the robot very small distances in the x direction to the left and to the right, and by combining these readings we can determine the edge location within Δx tolerance. Another way to determine the edge location is by rotating the robot clockwise and counter clockwise using small angles, and combining the readings as before to determine the edge location.

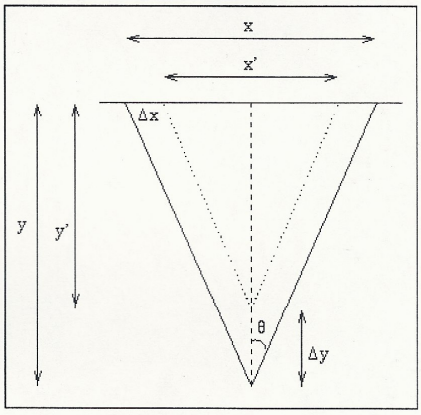

Figure 2: Translation in the y direction

Translation in the x direction

Moving the robot in the x direction will result in an overlapping region equals to:

$$y\tan\theta - 2\Delta x$$

as shown in Figure 4. The time needed for this movement is equal to:

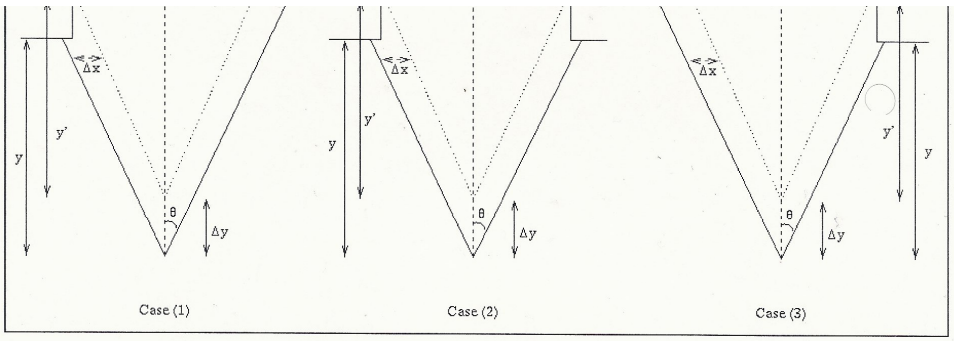

Figure 3: Different situations whilemovinh in the y direction

$$t_x = \frac{\Delta x}{v_r}$$

However, in our experiment, the robot can only move forward and backward. To move in the x direction, we need to rotate the robot 90 degrees, then move forwardΔx, and finally rotate back 90 degrees again. Therefore, the time needed to movΔx is:

$$t_x = \pi\omega + \frac{\Delta x}{v_r}$$

and by adding the time to decide taking this movement, the total time will be:

$$t = t_x + t_d$$

If the two readings are the same (i.e., y' = y), then we have two possibilities; the measured point is in the overlapping region, or it is outside the overlapping regionFigure 5 shows the two cases.

The probability that the two readings correspond to points in the overlapping region is:

$$\frac{y\tan\theta - 2\Delta x}{y\tan\theta + \Delta x}$$

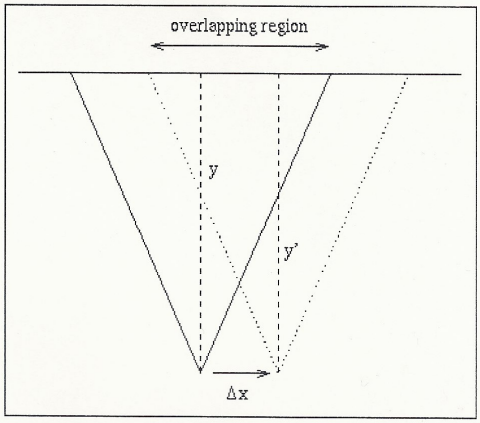

Figure 4: Translation in the x direction

This formula shows that the probability of the point to be in the overlapping region decreases by increasinΔx

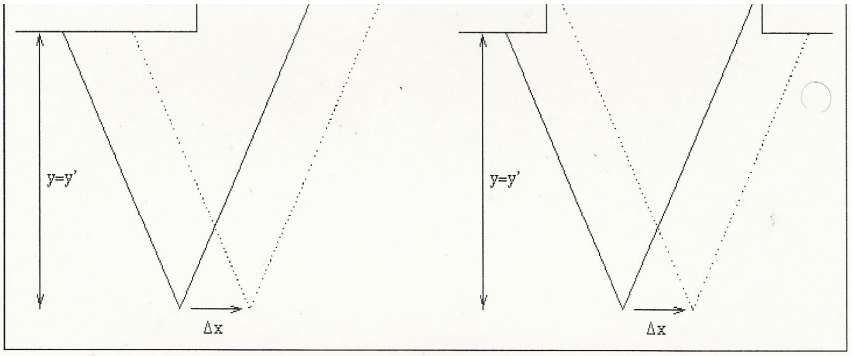

Figure 5: Moving in the x direction with y'=y

If the two readings are different (i.e., y≠ y), then again, we have two cases as shown in Figure 6; y' > y which means that there is an edge within the left Δx region, and y' < y which means that there is an edge within the right Δx region.

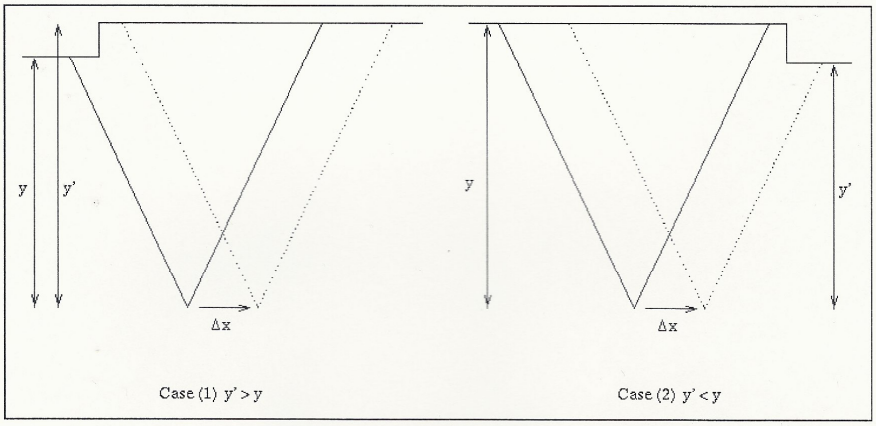

Figure 6: Moving in the x direction with y≠y

Rotation is similar to moving in the x direction. By rotating a small angle ϕ, where:

$$-2\theta < \phi < 2\theta$$

and with rotation radius r, there will be an overlapping area as shown in Figure 7. This overlapping area starts at a distance d_o which can be calculated as follows:

$$d_o = ec - gc$$

where

$$ec = r\sin(\phi)\tan(\phi + \frac{\pi}{2} - \theta)$$

$$gc = r - r\cos(\phi) = r(1 - \cos(\phi))$$

Finally, we have:

$$d_o = r[\sin(\phi)\tan(\phi + \frac{\pi}{2} - \theta) + \cos(\phi) - 1$$

For small values of ϕ, d_o can be approximated by:

$$d_o = \frac{r\phi}{\tan(\theta)}$$

Therefore, if the sensor reading is y, we should rotate the robot such that $y > d_o$, otherwise, the reading will be outside the overlapping region. In other words, the rotation angle ϕ is limited by the sensor reading as follow:

$$\phi < \frac{y\tan(\theta)}{r}$$

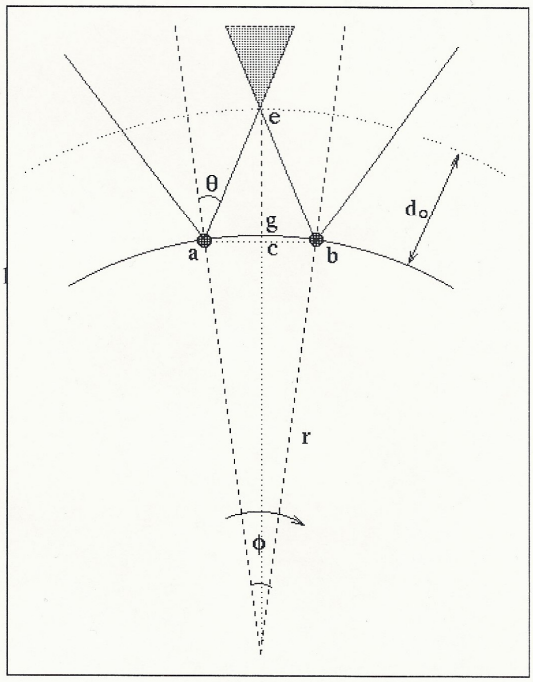

Figure 7: Rotating the robot ϕ degrees

Using case analysis similar to what we did for moving the robot in the x direction, with substituting Δx with $r\phi$, we can get new tolerance regions with probabilities associated to them in the same way we did before for the translation in the x direction.

The time needed to rotate ϕ degrees t_{rot} is equal to $\omega\phi$ and adding the decision time t_d we get the total time.

This case is exactly the same as rotating the robot, except for the fact that the angle θ is fixed. In case where the sensors are arranged in a circle and distributed on equal spacing angles, ϕ depends on the number of sensors used. For example, if we have 24 sensors, then $\phi = 2\pi/24$. To have an overlapping region, ϕ should be less than θ. Also, to consider this overlapping region, the sensor readings y for both sensors should be grater than d_o as discussed before.

Again, the case analysis that we did for translation in the x direction can be used here by replacing Δx with $r\phi$. This way we can get smaller tolerance areas with certain probabilities. The probability that the reading is in the overlapping area depends on the value of the readings and on the angles ϕ and θ.

SIMULATION RESULTS

In this experiment, We demonstrate the use of the tolerance measures discussed earlier. This experiment also illustrates the use of the logical sensors concept to implement high-level requests which incorporate tolerance measures and time calculations.

The request which was implemented for this experiment is *measure* which has the following syntax: ***measure (tolerance, time, preference)***, where *tolerance* is the required tolerance with 0 meaning get the best tolerance, and -1 means tolerance is not important. *time* is the required response time, and again 0 means as fast as possible, and -1 means time is not important. When both, *time* and *tolerance* are specified, the logical sensor may not be able to satisfy both criteria, and this is when *preference* is used to specify which criteria should be preferred. This request returns the resulting tolerance and the time consumed into the same parameters that were sent.

The following is the output of a program which uses this request to measure a point in front of the robot. First it sends a request to get the measure as fast as possible ignoring the tolerance.

```
Fast response required ...
!!! minimum time ...
!!!  current reading is 2071 mm, with tolerance 402.4 in time 0.9 msec.
Result: distance = 2071 mm, tolerance = 402.4, and time = 0.9 msec.
```

Second, the program sends a request to get the best accuracy (minimum tolerance), and the time is irrelevant.

```
Best tolerance required ...
!!! minimize the tolerance ...
!!!  current reading is 1536 mm, with tolerance 298.4 mm. in time 0.6 msec.
!!!  current reading is 1412 mm, with tolerance 274.3 mm. in time 2.5 msec.
!!!  current reading is 1383 mm, with tolerance 268.7 mm. in time 3.4 msec.
!!!  current reading is 1350 mm, with tolerance 262.3 mm. in time 4.4 msec.
!!!  current reading is 1291 mm, with tolerance 250.8 mm. in time 5.6 msec.
!!!  current reading is 1259 mm, with tolerance 244.6 mm. in time 6.5 msec.
```

```
!!!  current reading is 1168 mm, with tolerance 226.9 mm. in time 8.7 msec.
!!!  current reading is 1139 mm, with tolerance 221.3 mm. in time 9.6 msec.
!!!  current reading is 1091 mm, with tolerance 212.0 mm. in time 10.4 msec.
!!!  current reading is 1062 mm, with tolerance 206.3 mm. in time 11.0 msec.
!!!  current reading is 1015 mm, with tolerance 197.2 mm. in time 11.8 msec.
!!!  current reading is 971 mm, with tolerance 188.6 mm. in time 12.5 msec.
!!!  current reading is 938 mm, with tolerance 182.2 mm. in time 13.2 msec.
!!!  current reading is 894 mm, with tolerance 173.7 mm. in time 13.9 msec.
!!!  current reading is 847 mm, with tolerance 164.6 mm. in time 14.7 msec.
!!!  current reading is 818 mm, with tolerance 158.9 mm. in time 15.3 msec.
!!!  current reading is 756 mm, with tolerance 146.9 mm. in time 16.3 msec.
!!!  current reading is 724 mm, with tolerance 140.7 mm. in time 16.9 msec.
!!!  current reading is 694 mm, with tolerance 134.8 mm. in time 17.5 msec.
!!!  current reading is 633 mm, with tolerance 123.0 mm. in time 18.4 msec.
!!!  current reading is 603 mm, with tolerance 117.2 mm. in time 19.0 msec.
distance = 1536 mm,  tolerance = 117.2, and time = 19.0 msec.
```

Finally, the program specifies both time and tolerance to be met, preferring the time.

```
Tolerance required = 150.0, time required = 6.0 msec.
!!! both criteria are specified ...
!!!  current reading is 2190 mm, with tolerance 425.5 mm. and time 0.9 msec.
!!!  current reading is 2101 mm, with tolerance 408.2 mm. and time 2.7 msec.
!!!  current reading is 2068 mm, with tolerance 401.8 mm. and time 4.1 msec.
!!!  current reading is 2026 mm, with tolerance 393.6 mm. and time 5.6 msec.
```

Result: distance = 2190 mm, tolerance = 393.6, and time = 5.6 msec.

Figure 8 shows the movement of the robot while taking these measurements. The first request did not cause any movement since it required minimum time. The second request caused the robot to move forward to minimize the tolerance region. During this movement, the speed of the robot decreases to get better accuracy. Finally, the last request also caused the robot to move forward, but it stopped before reaching the required tolerance since the time was preferred.

In this experiment we used only the translation in the y direction to minimize the tolerance. However, the other two approaches mentioned earlier could be used to get better results with probability measures as well.

CONCLUSION

In this paper, tolerance measures for sonar sensors were proposed and different strategies to increase position accuracy were investigated. An example for applying this control scheme to a mobile robot was described along with the simulation results. We believe that this scheme provides more flexible and robust control systems, and allows more modular design for the whole control system. It also provides fast response for reaction behavior which is an essential requirement in real-time systems.

The next step to this work is to implement a distributed controller including higher level functions for increasing the accuracy of the measured point locations. These functions will be defined using the different tolerance analysis approaches discussed in the paper.

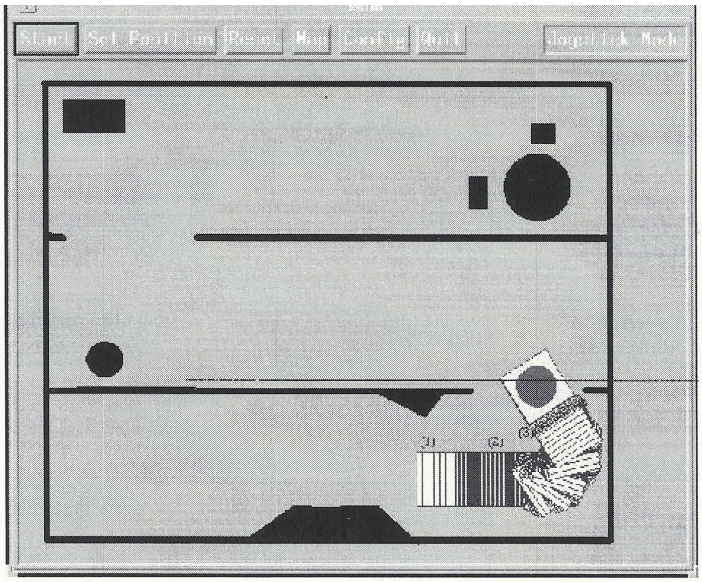

Figure 8: The trajectory of the robot while performing the requests

REFERENCES

1. M. Dekhil, T. M. Sobh, and A. Efros, "Commanding sensors and the control of mobile robots." IEEE Int. Conference. On Control Applications, Michigan, September 1996.
2. M. Dekhil, T. M. Sobh, and A. Efros, "Prototyping hybrid distributed control scheme for sonar-based mobile robots." Invited paper, the 10th IEEE Int. Symposium on Intelligent Control (ISIC 95), Monterey, California, August 1995.
3. R. R. Brooks and S. S. Iyengar, "Averaging algorithm for multi-dimensional redundant sensor arrays: solving sensor inconsistencies." Tech. Report, Louisiana State Univ. 1993.
4. L. Prasad, et. al., "Functional characterization in fault tolerant integrated sensor networks." IEEE Trans. Systems, Man, and Cybernetics, September 1991, pp. 1080-1087.
5. J. Budenske and M. Gini, "Why is it difficult for a robot to pass through a doorway using ultrasonic sensors?" In IEEE Int. Conference of Robotics and Automation, May 1994, pp. 3124-3129.
6. L. Kleeman and R. Kuc, "An optimal sonar array for target localization and classification." In IEEE Int. Conference. of Robotics and Automation, May 1994, pp. 3130-3135.
7. L. Korba, "Variable aperture sonar for mobile robots." IEEE Int. Conference. of Robotics and Automation, May 1994, pp. 3142-3147.