Sonar based localization for the PSUBOT: a computer assisted motorized wheelchair for the disabled capable of operating in a mapped environment.

Kevin Stanton[†]

Portland State University
Department of Electrical Engineering
PO Box 751
Portland, OR 97207
(503) 725-3807

find a new vector $\mathbf{t}'$ such that

$$\sum_{i=1}^{n}\mathbf{q}_i = \sum_{i=1}^{n}R_o(\theta)(\mathbf{p}_i-\mathbf{R})+(\mathbf{R}+\mathbf{t}')$$

where **R** is the current position of the robot before correction. Subtracting the above equations and rearranging yields the expression for $\mathbf{t}'$:

$$\mathbf{t}' = \mathbf{t}-R(\theta)(\mathbf{c}-\mathbf{R})+(\mathbf{c}-\mathbf{R})$$

The updated posture of the robot can then be found:

$$P(n)=(x_{n-1}+\mathbf{t}'_x, y_{n-1}+\mathbf{t}'_y, \theta_{n-1}+\theta)$$

9.3.2. Minimize Squared Error

We are now ready to derive expressions for the correction $(\mathbf{t},\theta)$ which, when applied to the points of the sonar scan image, matches them to the line segment model in the mean squared sense.

Define *S* as the sum of the squared distances between the points, each rotated about the centroid and translated by **t**, and the target lines (not line segments):

$$S=\sum_{i=1}^{n}\left([R_o(\theta)(\mathbf{p}_i-\mathbf{c})+(\mathbf{c}+\mathbf{t})]^t\mathbf{u}_i-r_i\right)^2$$

where there are *n* scan points $\mathbf{p}_i$ in the image whose centroid is **c**. $\mathbf{u}_i$ and r_i represent the target line of the point $\mathbf{p}_i$, and $R_o(\theta)$ is the rotation matrix.

Removing the $\sum$ and the square operator, the above equation can be seen in the form:

$$\text{Distance} = (\mathbf{P}^t\cdot\mathbf{u}-r)$$

where **P** is the point **p** rotated by θ about the centroid **c** and then displaced by the translation vector **t**. Since *S* would be linear without the trigonometric functions of $R_o(\theta)$, the pseudo-rotation matrix $R_p(\theta)$ is used to simplify the solution. However, since $R_p(\theta)$ is only an approximation of the true congruence, it will be necessary to iterate for the solution.

Setting the partial derivatives to zero:

$$\frac{\partial\mathbf{t}_x}{\partial S}=0 \qquad \frac{\partial\mathbf{t}_y}{\partial S}=0 \qquad \frac{\partial\theta}{\partial S}=0$$

yields a third order system of linear equations of the form **Ax** = **B**:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} \mathbf{t}_x \\ \mathbf{t}_y \\ \theta \end{bmatrix} = \begin{bmatrix} B_{11} \\ B_{21} \\ B_{31} \end{bmatrix}$$

where

A =

$$\begin{vmatrix} \sum_{i=1}^{n} \mathbf{u}_{i_x}{}^2 & \sum_{i=1}^{n} \mathbf{u}_{i_x}\mathbf{u}_{i_y} & \sum_{i=1}^{n} \mathbf{c}_y\mathbf{u}_{i_x}{}^2 - \mathbf{c}_x\mathbf{u}_{i_x}\mathbf{u}_{i_y} + \mathbf{u}_{i_x}\mathbf{u}_{i_y}\mathbf{v}_{i_x} - \mathbf{u}_{i_x}{}^2\mathbf{v}_{i_y} \\ \sum_{i=1}^{n} \mathbf{u}_{i_x}\mathbf{u}_{i_y} & \sum_{i=1}^{n} \mathbf{u}_{i_y}{}^2 & \sum_{i=1}^{n} \mathbf{c}_y\mathbf{u}_{i_x}\mathbf{u}_{i_y}\mathbf{c}_x\mathbf{u}_{i_y}{}^2 + \mathbf{u}_{i_y}{}^2\mathbf{v}_{i_x} - \mathbf{u}_{i_x}\mathbf{u}_{i_y}\mathbf{v}_{i_y} \\ \sum_{i=1}^{n} \mathbf{c}_y\mathbf{u}_{i_x}{}^2 - \mathbf{c}_x\mathbf{u}_{i_x}\mathbf{u}_{i_y} + \mathbf{u}_{i_x}\mathbf{u}_{i_y}\mathbf{v}_{i_x} - \mathbf{v}_{i_y}\mathbf{u}_{i_x}{}^2 & \sum_{i=1}^{n} \mathbf{c}_y\mathbf{u}_{i_x}\mathbf{u}_{i_y} - \mathbf{c}_x\mathbf{u}_{i_y}{}^2 + \mathbf{u}_{i_y}{}^2\mathbf{v}_{i_x} - \mathbf{u}_{i_x}\mathbf{u}_{i_y}\mathbf{v}_{i_y} & \sum_{i=1}^{n} (\mathbf{c}_y\mathbf{u}_{i_x} - \mathbf{c}_x\mathbf{u}_{i_y} + \mathbf{u}_{i_y}\mathbf{v}_{i_x} - \mathbf{u}_{i_x}\mathbf{v}_{i_y})^2 \end{vmatrix}$$

B =

$$\begin{bmatrix} \sum_{i=1}^{n} r\mathbf{u}_{i_x} - \mathbf{u}_{i_x}{}^2\mathbf{v}_{i_x} - \mathbf{u}_{i_x}\mathbf{u}_{i_y}\mathbf{v}_{i_y} \\ \sum_{i=1}^{n} r\mathbf{u}_{i_y} - \mathbf{u}_{i_x}\mathbf{u}_{i_y}\mathbf{v}_{i_x} - \mathbf{u}_{i_y}{}^2\mathbf{v}_{i_y} \\ \sum_{i=1}^{n} (\mathbf{u}_{i_x}\mathbf{v}_{i_y} - \mathbf{u}_{i_y}\mathbf{v}_{i_x} + \mathbf{c}_x\mathbf{u}_{i_y} - \mathbf{c}_y\mathbf{u}_{i_x}) \cdot (\mathbf{u}_{i_x}\mathbf{v}_{i_x} + \mathbf{u}_{i_y}\mathbf{v}_{i_y} - r) \end{bmatrix}$$

Gaussian elimination produces the following results:

$$\mathbf{t}_x = \frac{-A_{23}\cdot A_{32}\cdot B_{11} + A_{22}\cdot A_{33}\cdot B_{11} + A_{13}\cdot A_{32}\cdot B_{21} - A_{12}\cdot A_{33}\cdot B_{21} - A_{13}\cdot A_{22}\cdot B_{31} + A_{12}\cdot A_{23}\cdot B_{31}}{-A_{13}\cdot A_{22}\cdot A_{31} + A_{12}\cdot A_{23}\cdot A_{31} + A_{13}\cdot A_{21}\cdot A_{32} - A_{11}\cdot A_{23}\cdot A_{32} - A_{12}\cdot A_{21}\cdot A_{33} + A_{11}\cdot A_{22}\cdot A_{33}}$$

$$\mathbf{t}_y = \frac{A_{23}\cdot A_{31}\cdot B_{11} - A_{21}\cdot A_{33}\cdot B_{11} - A_{13}\cdot A_{31}\cdot B_{21} + A_{11}\cdot A_{33}\cdot B_{21} + A_{13}\cdot A_{21}\cdot B_{31} - A_{11}\cdot A_{23}\cdot B_{31}}{-A_{13}\cdot A_{22}\cdot A_{31} + A_{12}\cdot A_{23}\cdot A_{31} + A_{13}\cdot A_{21}\cdot A_{32} - A_{11}\cdot A_{23}\cdot A_{32} - A_{12}\cdot A_{21}\cdot A_{33} + A_{11}\cdot A_{22}\cdot A_{33}}$$

$$\theta = \frac{-A_{22}\cdot A_{31}\cdot B_{11} + A_{21}\cdot A_{32}\cdot B_{11} + A_{12}\cdot A_{31}\cdot B_{21} - A_{11}\cdot A_{32}\cdot B_{21} - A_{12}\cdot A_{21}\cdot B_{31} + A_{11}\cdot A_{22}\cdot B_{31}}{-A_{13}\cdot A_{22}\cdot A_{31} + A_{12}\cdot A_{23}\cdot A_{31} + A_{13}\cdot A_{21}\cdot A_{32} - A_{11}\cdot A_{23}\cdot A_{32} - A_{12}\cdot A_{21}\cdot A_{33} + A_{11}\cdot A_{22}\cdot A_{33}}$$

We now review the matching process:

1. A SONAR scan produces a set of data points in polar form (d_i, α_i) which are transformed into the Cartesian world coordinates - $\mathbf{p}_i = (x_i, y_i)$ - based on the believed posture of the robot (the posture which is to be corrected).
2. A target line L_j is assigned to each point p_i by finding the line segment l_j whose distance from the point is minimum.
3. Repeat the following until $(\mathbf{t}, \theta)$ is very small.
    3.1 Find $(\mathbf{t}, \theta)$ which minimizes the mean squared distance between points and their targets using pseudo-rotation.

3.2 Apply $(\mathbf{t},\theta)$ to the SONAR data points using *true* rotation.

4. Add up all the corrections applied to the points under step 3. This is the solution vector $(\mathbf{t},\theta)$.
5. Find translation and rotation (based at the current position of the robot) which yields the same translation and rotation based at the centroid of SONAR data points.
6. Apply the solution vector to the old posture to calculate a new matcher-corrected posture of the robot.

9.4. Comments on the Matcher

The algorithm as described is subject to several downfalls due to the straight ahead mathematical approach, so let us take several steps back for a moment to expose these problems and their solutions.

9.4.1. Finding the correction vector

Notice that all correction vectors of step 3 are summed in step 4 to yield the final correction vector. We are able to do this because of the property of points rotated about their centroid. To translate and rotate a set of points by $\mathbf{t}_i,\theta_i$ about their centroid a number of times is equivalent to translating by $\sum\mathbf{t}_i$ and rotating by $\sum\theta_i$ This property is not true unless rotation occurs around the centroid of points.

9.4.2. Removal of Outliers

Due to the noisy nature of the SONAR range data [sonar is noisy reference], it was necessary to remove certain data points to allow closer matches. The rationale used here is that some points are not simply ‘‘inaccurate’’, but rather are wrong (the data doesn’t correspond to any physical distance we are interested in measuring). There are several causes for such extraneous data. One is the affect of echos, especially near corners where the sound waves may reflect off of several walls before returning to the transducer. Echoing leads to misleading readings of long distances and should be removed.

Another cause of erroneous range data is perhaps illustrated best with an example: Suppose the robot is in a corridor having a width of 8 feet and a height of 12 feet (not unlike the PCAT building in which testing occurred). Additionally assume that the SONAR device is pointing in the direction of a wall at the end of the corridor which is out of range. Clearly a finite distance reading will be incorrect, yet often occurs due to reflections of the ultrasonic sound wave off the walls, ceiling, and floor. Again, it is emphasized that we are not trying to remove data points which are slightly inaccurate, but those which have no meaningful physical relationship to the world.

The method employed by [Cox91a] is straight forward: remove any points whose distance from their target line segment is greater than some pre-defined constant. The assumption which makes the above method work is that of ‘small step’; the current estimated posture is *close* to the real posture. If the small step assumption were not true for a particular matching session, valid and possibly critical points could be removed while extraneous, noisy points would likely be retained. A more adaptive algorithm seemed to us a better choice.

We compute the average μ and standard deviation (σ) of the errors:

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\text{error}_i$$

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (\text{error}_i - \mu)^2$$

and remove those data points whose distance from their target line segment outside the range: ($mu - F\sigma$, $\mu + F\sigma$) where F is determined experimentally to reflect the fraction of erroneous data points. It would perhaps be possible to find some correlation between the shape of the current room (line segment world model) and this quantity. We have not implemented such a scheme, but use the constant value of $F = 2$. This value of F retains about 95% of the points (assuming the distribution of the errors is Gaussian [Guttm82a].

9.4.3. Subtle implementation/intention discrepancy

Recall the meaning of the error term which is minimized by the matcher: The distance between a point and its *infinite* length target line. In other words, while the target of a point is determined by its distance from the line segments (as expected), the actual minimization equation effectively extends the target line segment to infinity in both directions and then finds the distance from the point to *this* line which causes points to be drawn toward *phantom* lines. The desired and calculated distances are different only if the distance from the point **p** to its target line segment *l* does not lie on the perpendicular through **p** to *L*. However, since target lines are determined based on distances to line *segments*, the affect of this undesirable side effect is reduced when outliers are removed. In a typical continuous line segment world model, such troublesome points occur mainly at convex (outside) corners. An algorithm could be imagined which would hunt down and eliminate these data points.

9.4.4. Indeterminate Results

The system matrix in section 3.3.2 is singular ($\frac{0}{0}$ results) when all target lines in the calculation of the correction vector are parallel, or nearly parallel. See Figure 20.

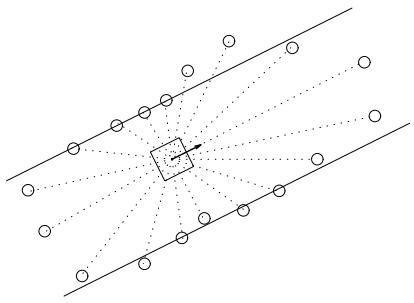

FIGURE 20.

Such a situation would not be uncommon with a robot traveling down a corridor. $\frac{0}{0}$ implies an infinite number of solutions. This makes intuitive sense when the robot is too far from the ends of a corridor to determine its lateral position. Clearly the matcher should be capable ot determining the correction perpendicular to the walls, but the position along the hallway cannot be determined from the available SONAR data points. Cox's solution to this problem is to make modifications to the system matrix and introduce predefined, experimentally determined constants to increase the system stability. Again we attempt a more

rigorous approach. Our modification of the procedure is the topic of the next section.

IV. Modifications for Parallel Target Lines

As stated earlier, an assumption which is made is that most points are near the line segment of the model they represent in the real world. This is generally a valid assumption since the purpose of the sonar matcher is to provide an update to the dead reckoning so the posture errors do not grow without bound as they would otherwise. However, even when the small step assumption is satisfied, the matching algorithm described in the previous section yields an infinite number of solutions if all the target liens are parallel. Consider a simple example shown in figure 21.

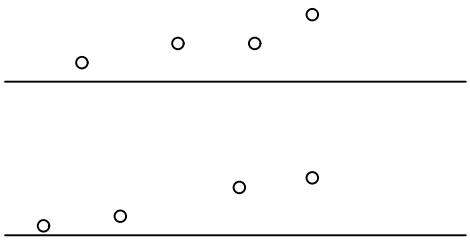

FIGURE 21.

Clearly the data points need to move in the negative *y* direction and rotate. Any value for *x*, however, yields a minimum error term and the system matrix *A* is found to be singular.

We identify three reasons why such a situation could appear in practice:

1. Data points are distant from the model (the robot is lost). In this case there is often one closest line segment which is the target of all the points.
2. The robot is traveling down a corridor: Long, narrow, parallel line segments are the targets of the points.
3. Only a partial scan was taken. If the robot were following a wall in a very large room, it may decide to acquire data only where it expects to find it valid. As in 1. above, all data points would likely have the same line segment as their target.

Since **t** in the solution vector is under-specified, we reduce the dimension of the problem from three to two; i.e. we require the data points to move only perpendicular to their target lines. Rotation, of course, is also performed.

The vector **v** is assigned to be perpendicular to the (parallel) target lines, and *K* becomes the multiplier by which the points are moved along **v**. The sum of the least squares equation becomes:

$$S=\sum_{i=1}^{n}\left([R(\theta)(\mathbf{p}_i-\mathbf{c})+(\mathbf{c}+K\mathbf{v})]^t\mathbf{u}_i-r_i\right)^2$$

Following the same procedure as before, we find the values for *K* and θ which minimizes *S* by setting the partial derivatives to zero:

$$\frac{\partial K}{\partial S}=0 \qquad\qquad \frac{\partial \theta}{\partial S}=0$$

and solving the second order system of linear equations:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} K \\ \theta \end{bmatrix} = \begin{bmatrix} B_{11} \\ B_{21} \end{bmatrix}$$

where

A =

$$\begin{vmatrix} \sum_{i=1}^{n} (\mathbf{u}_{i_x} px + \mathbf{u}_{i_y} py)^2 & \sum_{i=1}^{n} (px\mathbf{u}_{i_x} + py\mathbf{u}_{i_y})(\mathbf{c}_y \mathbf{u}_{i_x} - \mathbf{c}_x \mathbf{u}_{i_y} + \mathbf{u}_{i_y} \mathbf{v}_{i_x} - \mathbf{u}_{i_x} \mathbf{v}_{i_y} \\ \sum_{i=1}^{n} (\mathbf{u}_{i_x} px + \mathbf{u}_{i_y} py)(-\mathbf{u}_{i_x} \mathbf{v}_{i_y} + \mathbf{u}_{i_x} \mathbf{c}_y + \mathbf{u}_{i_y} \mathbf{v}_{i_x} - \mathbf{u}_{i_y} \mathbf{c}_x) & \sum_{i=1}^{n} (-ux\mathbf{v}_{i_y} + \mathbf{u}_{i_x} \mathbf{c}_y + \mathbf{u}_{i_y} \mathbf{v}_{i_x} - \mathbf{u}_{i_y} \mathbf{c}_x)^2 \end{vmatrix}$$

B =

$$\begin{vmatrix} -(px\mathbf{u}_{i_x} + py\mathbf{u}_{i_y})(-r + \mathbf{u}_{i_x} \mathbf{v}_{i_x} + \mathbf{u}_{i_y} \mathbf{v}_{i_y}) \\ -(\mathbf{u}_{i_x} \mathbf{v}_{i_x} + \mathbf{u}_{i_y} \mathbf{v}_{i_y} - r)(-\mathbf{u}_{i_x} \mathbf{v}_{i_y} + \mathbf{u}_{i_x} \mathbf{c}_y + \mathbf{u}_{i_y} \mathbf{v}_{i_x} - \mathbf{u}_{i_y} \mathbf{c}_x) \end{vmatrix}$$

The solution vector may then be determined:

$$K = \frac{B_{21} - \frac{B_{11}}{A_{12}} A_{22}}{A_{21} - \frac{A_{11}}{A_{12}} A_{22}}$$

$$\theta = \frac{B_{21} - \frac{A_{21}(B_{21} - \frac{B_{11}}{A_{12}} A_{22})}{A_{21} - \frac{A_{11}}{A_{12}} A_{22}}}{A_{22}}$$

and the correction is:

$$P_n(n) = \left(x_{n-1} + K\mathbf{v}_x \, , \, y_{n-1} + K\mathbf{v}_y \, , \, \theta_{n-1} + \theta \right)$$

where the subscript of *n* denotes the current matching session.

If all target lines are found to be parallel, matcher2 is used. However, since the problem may be of type (1) above, moving the data points closer to the model may make the ‘small step’ assumption valid as the targets become non-parallel. It is for this reason that we test for an impending singularity (parallel targets) before each iteration of the algorithm.

V. Experiment

The formal testing of the sonar matching algorithm was accomplished by mapping a room, gathering sonar data, corrupting it, applying the matcher, and analyzing the results. From these results we hoped to characterize the performance of the matching algorithm under different conditions, in different rooms, and under bounded but random odometry errors.

Figure 22 illustrates the steps used to test the matcher algorithm.

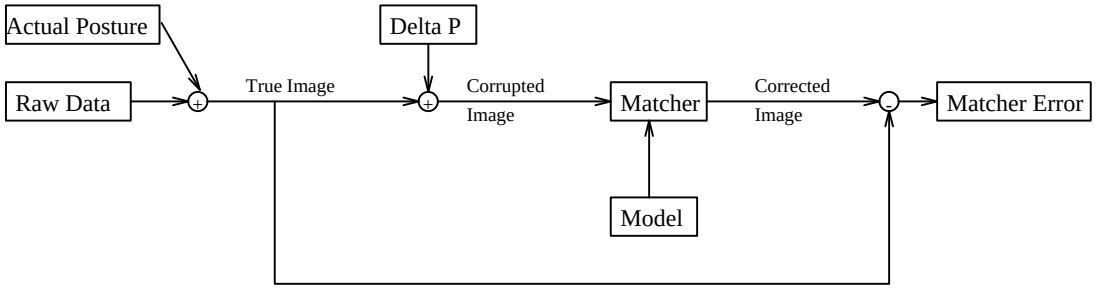

FIGURE 22.

First, a blueprint was secured from the architecture department with the physical plant of PSU. The origin was arbitrarily chosen as the NW corner of the building with zero degrees pointing south. Next the walls of a section of the building was turned into line segments by finding the coordinates of their end-points. The segments representing a room is called the model. It was found that some walls on the blueprint were inaccurate and others appeared to be simply in the wrong place. This showed that with a building with many temporary partitions and multiple reconstructions, drawing cannot be relied upon wholeheartedly and must be verified using a tape measure and a friend.

Since the scope of this work did not include generating, searching, or retrieving line segment models from the 'world database', applicable lines were manually chosen and entered into a file for each scan location.

The next step was to move the robot to a known location and perform a sonar scan. The location is called the 'actual posture'. These scans were then transferred from the PC to the Unix system for analysis.

To analyze the algorithm we needed to induce positional errors into the data to see how the matcher performed in spite of them. Since all the points of a particular scan were taken from the same place (taken without the robot moving), a simple shift and rotation of the image did just that. The amount of shift in x and y and the amount of rotation (θ) about the robot's location is called the induced Δ 'Delta'. We define its components as: $\Delta P = (\Delta x, \Delta y)$ and the change in θ as $\Delta\theta$.

The matching algorithm may then be employed on the corrupted data in an attempt to negate the induced Δ. The difference between the result of the matcher and the actual posture is called the 'error' of that matching session. It should be noted that a Δ of (0,0,0) does not, in general, yield an error or (0,0,0).

Figure 23 shows a ΔP grid with an assigned $\Delta\theta$ of $20^{(de}$ and how an image is corrupted (shifted and rotated) by adding a particular ΔP and rotating by $\Delta\theta$.

Matching each corrupted image from each setting of each room produces a seeming myriad of relationships between which relationships could be shown. For example: How does the Δx (or Δy or $\Delta\theta$) affect the ability of the matcher to converge; how does the number (or closeness or length or one-

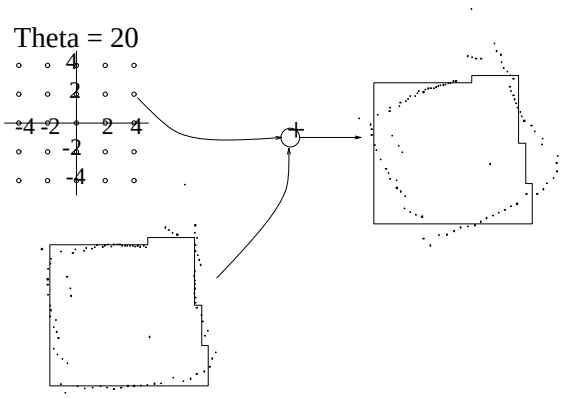

FIGURE 23.

sidedness) of the lines affect the convergence (or average x,y or θ errors) of the matcher; how does the change in the parameter F in the noise-removal module affect the convergence (or x,y, or θ errors) of the matcher; and so on. Worse yet, once we tabulated and graphed these relationships, how would we interpret the results? Fundamentally, however, the question at hand is whether the matcher is capable of keeping the robot on course; its ability to converge on the correct minimum in the presence of odometry-based errors.

So to draw conclusions we analyze the data using two methods: We look at the maximum and average errors versus the arc length of ΔP for various $\Delta\theta$'s; and we look to see where the algorithm converges in a particular room for various $\Delta\theta$'s. Finally, we analyze the complexity of the algorithm and perform some benchmarks.

The final and ultimate test of the matcher is to combine it with a pilot algorithm on the robot and observe how it maneuvers within the building using its odometry and occasional matching for navigation.

Expected Results

Since the matching algorithm works by converging on the nearest minimum sum of mean squared errors, we expected to see a small final error when inducing small ΔP's, and large errors for large ΔP's. The relationship was not expected to be linear but binary: either the matcher converges on the correct local minimum or it converges on the wrong one (yielding a larger error). The expected graph of absolute error versus ΔP is shown in figure 24. Actual results are shown in a later section.

Types of Rooms

Four types of rooms were used in our analysis of the matcher: small rectangular, large rectangular, corridor (two parallel walls), and a combination of the above. Each type has associated good and bad points which are discussed below.

A small rectangular room is one in which the matcher performs the best. Its small size allows the sonar device to locate all four walls, yielding a well-founded match. It was found that the matcher often converged correctly despite very large translations. A problem with rectangular rooms is that a rotation of the image by 45° or more often caused the matcher to converge on a correction whose heading was off by

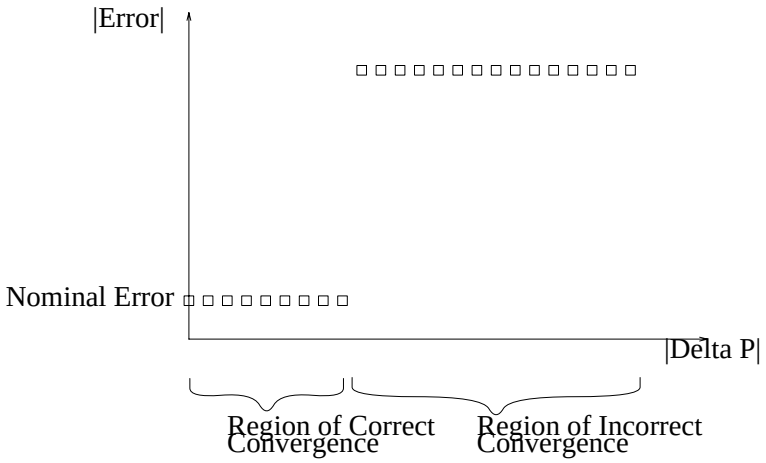

FIGURE 24. Error Versus Delta P

90°.

Large rectangular rooms present the additional problem of range. If walls are too far to be reached by the sonar, somewhat shorter distances are recorded which must be removed as noise. Also, if only one wall is found with the sonar, the distance along the wall remains unknown until a non-parallel wall is reached. This is the problem of parallel (single) target lines as seen in the next room type. Another more subtle characteristic of scans taken in larger rooms is that a small error in heading moves distant readings proportionally far from their true location. The result is lower rotational noise immunity.

Parallel walls present a situation where the position along the hall is non-deterministic and is assumed to be the position given by the odometry. However, $\Delta\theta$ may be as large as 90° without an incorrect convergence.

Intersecting corridors and other complex rooms are perhaps the most difficult to match due to the fact that a small ΔP can place points close to the wrong side of either or both hallways leading to an incorrect convergence. In other words, there are many false local minimums near the correct one when many mutually close parallel lines make up the local line segment model.

One coule certainly produce other room types from which additional conclusions could be drawn, but we chose to limit our observations to the simple-yet-complete types shown for simplicity. We also believe that these rooms are characteristic of those most often occuring in ‘real’ buildings. This is especially true of corridors through which the robot would spend much of its time navigating.

Analysis of Matching Errors

We begin our presentation of the results of the simulations by using several concrete examples from which we can begin to evaluate the matching algorithm in practice. All four types of rooms are represented with various Δ’s used.

10. Room 0

10.1. Correct Convergence

We begin with a small rectangular room. This is the basement lab of Dr. Perkowski's in which much of the PSUBOT was developed. The room has shelves covering nearly 3 of the walls and other assorted clutter which was not entered into the model of the room. The wall on the left is smooth, and in the scan shown, I am standing near the doorway which is near the right side of the bottom wall. First we show the results of the matcher with Δ = (-5,-5,0.35); i.e. the estimated position is wrong by 5 feet in both *x* and *y* directions, and the believed heading is wrong by 20°. See following three figures.

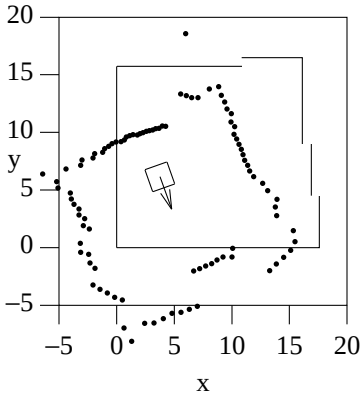

FIGURE 25. Shifted and Rotated Image

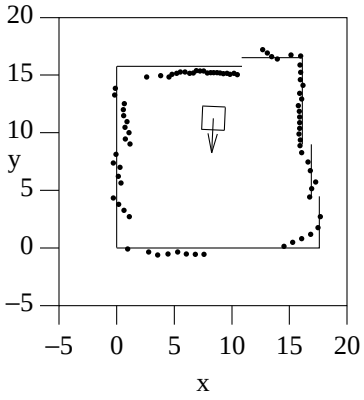

FIGURE 26. Matched Image Without Noise

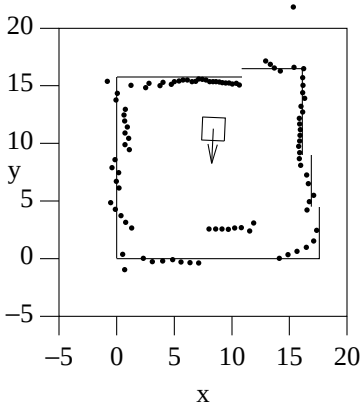

FIGURE 27. Final Matched Image

The second image shows the data considered just before the last correction. From it we can see that the noise removal algorithm was successful in eliminating those points representing the person standing near the wall. Table XXX tabulates the statistics of the matching session shown.

We can see from the table that initially there were many points dropped, as there were many points far from their targets. As they came closer to their target lines the number was reduced to those which were truely noisy or simply wrong (for a distinction see Chapter IV/Removal Of Outliers). Notice also, that the process terminated when the maximum change in *x* and *y* was less than 0.2 feet and the maximum

$\Delta=(-5,-5,0.35)$, $F=1$

Iteration	Average Dist	Std Dev of Dist	Pts Dropped	Correction x	Correction y	Correction θ (rad)
1	-0.628255	5.74415	36	2.72101	2.13262	0.01138
2	0.894017	4.52347	31	1.12595	1.20243	-0.150398
3	1.25128	4.64336	29	0.57043	1.05755	-0.115745
4	1.27601	4.20551	19	0.206124	0.571806	-0.101845
5	1.31315	3.90068	15	0.026033	0.133716	-0.041724
Total				4.64955	5.09812	-0.398332
Error				-0.350452	0.098124	-0.048332

change in θ was less than 0.1. Faster matching sessions may be achieved by changing the termination condition at an expense of accuracy.

As can be seen both pictorially and numerically the matching algorithm found the correct orientation of the robot dispite the initial error. shown.

10.2. Incorrect Convergence

We now demonstrate a situation where the matcher converges on the wrong local minimum. If $\Delta=(5,-5,0.35)$ the matcher continues to rotate the image further rather than rotate it back. The result is a relatively good (yet incorrect) match for this rather square-like room as it is similar after rotating 90°. Also notice that it takes many more iterations for the algorithm to settle. This is because of the psudo-rotation used in the derivation of the sum of least squares equations. It is found that large rotations require many iterations.

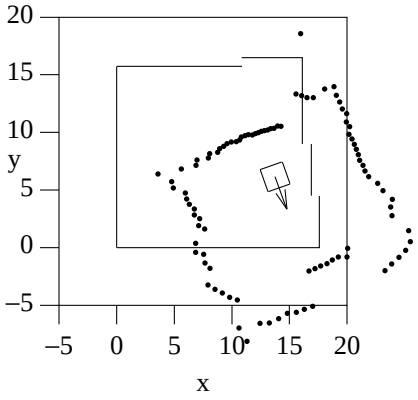

FIGURE 28. Shifted and Rotated Image

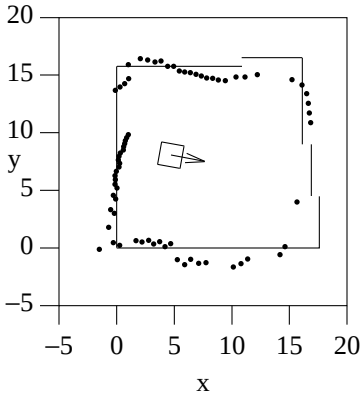

FIGURE 29. Matched Image Without Noise

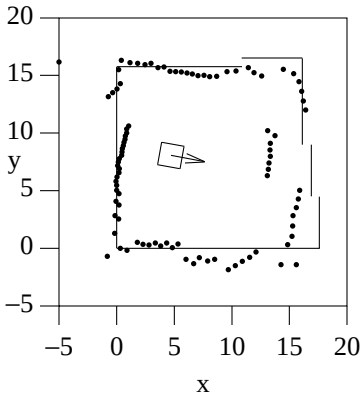

FIGURE 30. Final Matched Image

Looking at the two matching sessions we can see that for this image the matcher worked for $\Delta=(-5,-5,0.35)$ but did not work for $\Delta=(5,-5,-.35)$. From this we see that convergence depends somewhat on the direction of displacement and not on its magnitude only (the magnitudes were exactly the same).

We forgo showing incorrect matchings for the remainder of the rooms. Our purpose in this section is to give the reader a feel for what types of images can be expected from various room, not to show every possible matching session.

$\Delta=(5,-5,0.35)$, $F=1$

Iteration	Average Dist	Std Dev of Dist	Pts Dropped	Correction x	Correction y	Correction θ (rad)
1	2.92486	6.31638	40	0.127534	0.802872	0.110878
2	3.94704	6.08292	37	-1.22332	-0.041724	0.023026
3	3.71266	6.52362	40	-1.11634	0.635029	0.054278
4	3.33529	6.69707	42	-1.11096	0.978665	0.103298
5	3.35244	6.13868	42	-0.624043	0.500646	0.042937
6	3.62995	5.56763	38	-0.591566	0.168663	0.058923
7	3.48851	5.82722	39	-0.395784	-0.001905	0.029393
8	3.18915	4.99423	34	-0.681239	-0.062811	0.051554
9	3.12735	5.51086	37	-0.65641	-0.123027	0.091193
10	2.44497	6.02809	37	-0.755652	-0.343665	0.090703
11	1.39015	6.05886	36	-0.665905	-0.429504	0.079256
12	0.962728	5.48436	33	-0.46129	-0.093625	0.05585
13	0.926138	5.21210	35	-0.299728	0.100147	0.047837
14	0.954267	5.13532	34	-0.500692	-0.179407	0.103302
15	0.023358	3.74207	25	-0.092678	-0.020519	0.094662
Total				-9.04808	1.88983	1.03709
Error				-4.04808	-3.11017	1.38709

11. Room 1

The second room demonstrated is a large rectangular room. It is the main lobby of the PCAT building. It is uncluttered except for some chairs which are in a line about 4 feet the walls on the right and left. There are also some short tables which do not enter into the image because of their height. The walls behind the chairs are nearly ideal for sonar. They have vertical slats spaced about one inch apart which allows the ultrasonic pulse to bounce back to the transmitter. As an added bonus corkboard lines the back side of the slats for a reduction in echo.

Figure 31 shows the robot near the double doors. Several people were watching as the scan was made, as can be seen from the apparent wall above and to the left of the robot. Notice that most of this noise was removed in the second image.

12. Room 2 Next we demonstrate the matcher in a corridor. To the matcher, there are only two parallel target lines, so it uses its parallel (two dimensional) solution. This means that the matcher will correct the position of the robot only along the corridor; in the y direction.

As this corridor is made up of glass on both sides, it proves to yield noisy scans. Since it is long and narrow and due to the spreading effect of the sonar beam, long distances along the hallway are nearly always shown as much shorter than they really are. The result is what looks like a very narrow rectangular room.

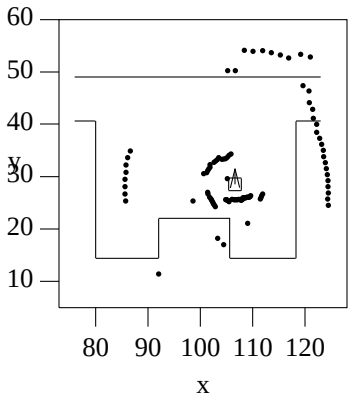

FIGURE 31. Large Rectangular Room

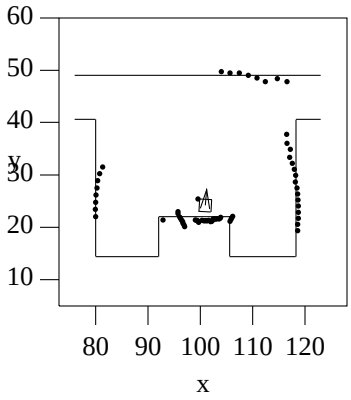

FIGURE 32. Matched Image Without Noise

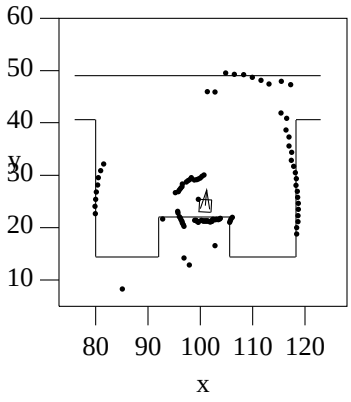

FIGURE 33. Final Matched Image

Also note that the algorithm is more tolerant of noise in this setting than in the previous ones. This is because the standard deviation of the distance from each point to its target line is much larger. However, that noise which remains after filtering tends to be cancelled out by a point from the opposite wall.

Here are the three graphs to demonstrate the matcher:

$\Delta=(6,4,0)$, $F=1$

Iteration	Average Dist	Std Dev of Dist	Pts Dropped	Correction x	Correction y	Correction θ (rad)
1	1.34258	12.0981	29	-5.62671	-4.11459	0.053811
2	-2.30444	12.1663	25	-0.007132	-0.195088	-0.101356
3	-1.92595	12.8647	25	-0.028048	-0.009131	-0.031843
Total				-5.66189	-4.31881	-0.079388
Error				0.338109	-0.318807	-0.079388

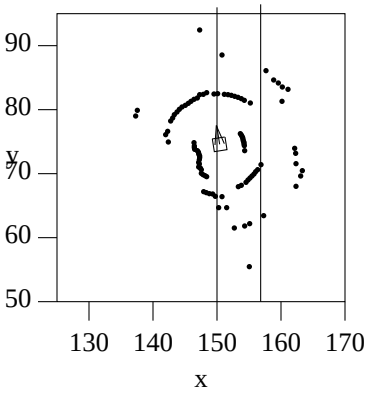

FIGURE 34. Large Rectangular Room

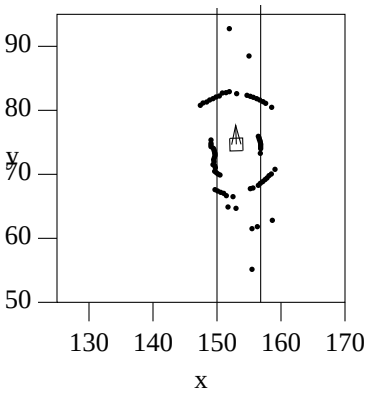

FIGURE 35. Matched Image Without Noise

Regions of Convergence

A large number of pictures and tables as shown above with varying Δ's would not be adequate in our attempt to characterize the matchers performance or to ensure with reasonable doubt that it would correct for reasonable odometry errors. So we chose to graph the region of convergence for various rooms. Such a graph would involve many matching sessions like those shown above, but with dots shown at those locations from which the matcher was able to converge. Relating this to the Δ grid shown previously, we show those points corresponding to the Δ's from which the final error is in near proximity to the error resulting from $\Delta P=(0,0)$. One could visualize it as a slice of the three-dimensional graph of the error versus ΔP.

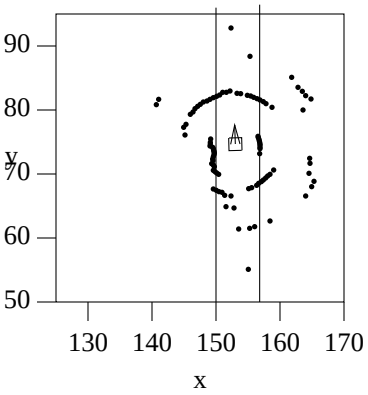

FIGURE 36. Final Matched Image

$\Delta=(-3,0,0.1745)$, $F=1$

Iteration	Average Dist	Std Dev of Dist	Pts Dropped	Correction x	Correction y	Correction θ (rad)
1	-2.27493	4.75576	27	2.02419	0.015018	-0.025477
2	-0.592822	3.94933	22	0.321128	0.020192	-0.033475
3	-0.398065	3.82609	21	0.228218	0.034148	-0.054835
4	-0.287945	3.79165	23	0.043171	0.013953	-0.021842
Total				2.61670	0.083311	-0.135628
Error				-0.383295	0.083311	0.038872

Since such a graph could be misleading - the matcher converged incorrectly even for $\Delta P=(0,0,0)$ - we include the error vector and angle with each diagram. The points are superimposed over the line-segment room and the robot's actual location is designated by a box. The actual heading of the robot can be seen by the solid arrow, and $\Delta\theta$ is shown as a dotted line.

References

Kerth88a.

Norman L. Kerth, ''MOOD: A Methodology for Structured Object Oriented Design,'' *Tutorial presented at OOPSLA 88, San Diego*, 1988.

Cox91a.

Ingemar J. Cox, ''Blanche-- An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle,'' *IEEE Transactions on Robotics and Automation*, pp. 193-204, April 1991.

Mayi91a.

Dieudonne Mayi, ''Using Computer Vision for the Robot Localization Problem,'' *Northcon 91*, October 1-3, 1991.

Lozan79a.

T. Lozano-Perez and Wesley, M. A., ''An Algorithm for Planning Collision-Free Paths Among

Polyhedral Obstacles,'' *Communications of the ACM, vol. 22, no. 10*, pp. 560-570, October 1979.

Lozan81a.

Lozano-Perez, T., ''Automatic Planning of Manipulator Transfer Movements,'' *IEEE Transactions on Systems, Man and Cybernetics*, vol. 11(10), pp. 681-698, 1981.

Elfes87a.

Alberto Elfes, ''A Sonar-Based Real-World Mapping and Navigation,'' *IEEE Journal of Robotics and Automation*, vol. RA-3, No. 3, pp. 249-265, 1987.

Khati86a.

Khatib, O., ''Real-Time Obstacle Avoidance for Manipulators and Mobile Robots,'' *International Journal of Robotics Research*, pp. 90-98, 1986.

Kodit89a.

D. E. Koditschek, ''Robot Planning and Control via Potential Functions,'' *The Robotics Review 1*, pp. 349-368, 1989.

O'Dú82a. O'Dúnlaing, C. and Yap, C.K., ''A Retraction Method for Planning the Motion of a Disc,'' *Journal of Algorithms*, vol. 6, pp. 104-111, 1982.

Latom91a.

Jean-Claude Latombe, in *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

Matar90a.

Maja J. Mataric, ''A Distributed Model for Mobile Robot Environment Learning and Navigation,'' MIT Technical Report #1228, May 1990.

Brooka.

Anita Flynn & Rodney Brooks, *Battling Reality*, MIT AI Memo 1148.

Flynn89a.

Anita M. Flynn, Rodney A. Brooks, William M. Wells III, and David S. Barrett, ''SQUIRT: The prototypical Mobile Robot for Autonomous Graduate Students,'' MIT AI Memo# 1120, July 1989.

Crowl89a.

Crowley, J.L., ''World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging,'' *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 674-680, 1989.

Lumel87a.

Vladimair J. Lumelsky and Alexander A. Stepanov, ''Path-Planning Stratevies for a Point Mobile Automation Moving Amidst Unknown Obstacles of Arbitrary Shape,'' in *Algorithmica*, Springer-Verlag, New York, 1987.

Espin1.a.

Cecilia Espinosa and Marek A. Perkowski, ''Hierarchical Hough Transform for the Vision System of a Wheelchair Robot,'' *Northcon 91*, October 1-3, 1991..

Evere89a.

CDR H. R. Everett, ''Survey of Collision Avoidance and Ranging Sensors for Mobile Robots,''

Robotics and Autonomous Systems, vol. 5, pp. 5-54, Elsevier Science Publishers B.V., North-Holland, 1989.

Elfes86a.

Alberto Elfes, ‘‘A Sonar-Based Mapping and Navigation System,’’ *Proc of 1986 IEEE International Conference on Robotics and Automation*, pp. 1151-1156, 1986.

Cox89a.

Ingemar J. Cox, ‘‘Blanche: Position Estimation for an Autonomous Robot Vehicle,’’ *Proceedings of the IEEE/RSJ International Workshop on Robots and Systems (IROS) ’89)*, pp. 432-439, Sept. 4-6, 1989.

Iijim81a.

J. Iijima, S. Yuta, and Y. Kanayama, ‘‘Elementary Functions of A Self-Contained Robot - YAMA-BICO 3.1,’’ *Proceedings of the 11th International Symposium on Industrial Robots*, pp. 211-218, Tokyo, 1981.

Wang88a.

C. Ming Wang, ‘‘Location Estimation and Uncertainty Analysis for Mobile Robots,’’ *International Conference on Robotics and Automation*, pp. 1230-1235, April 24-29, 1988.

Nelso88a.

Winston L. Nelson and Ingemar J. Cox, ‘‘Local path Control for an Autonomous Vehicle,’’ *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, pp. 1504-1510, 1988.

Guttm82a.

Irwin Guttman, S. S. Wilks, and J. Stuart Hunter, *Introductory Engineering Statistics*, Wiley Seriew in Probablity and Mathematical Statistics-Applied, John Wiley & Sons, 1982.