Autonomous Mobile Robot Self-Referencing with Sensor Windows and Neural Networks

Jason A. Janét, Ricardo Gutierrez-Osuna, Michael G. Kay and Ren C. Luo
Center for Robotics and Intelligent Machines
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27695-7911

Abstract *- When navigating an environment a mobile robot can update its position and orientation by searching known landmarks and compare predictions with observations. This paper presents a method of mobile-robot self-referencing where every mapped object (obstacles to the global motion planner) in the environment can be used as potential sources of position and orientation information. This approach employs the efficiency of traversability vectors (t-vectors) for finding in-range geometric beacons and isolating surfaces visible to a sensor. Configuration-space (C-space) buffering (growing polygons to keep motion a safe distance from objects) will reduce the search time for finding in-range geometric beacons. Finally, a small multi-layered neural network is used to provide a credence value for each predicted range that can be factored in to a filter or control strategy. This approach can be generalized to any ranging sensor that samples a region (e.g. IR sensors).*

1. Introduction

A self-referencing approach must be able to predict the most likely range from a surface *anywhere* inside a sensor's scan cone. Although the variables involved with reliably predicting a range are numerous, a few geometric properties can be used to calculate an expected time-of-flight (TOF) sensor reading. However, in some situations thresholded values of these geometric properties do not reflect the true behavior of the sensor. Hence, to refine the range prediction process, a multi-layered neural network is used to model the credibility of a predicted time-of-flight (TOF) reading. This *credence factor* is based on a combination of certain geometric properties. This paper will show that the neural network can also model the true boundaries of an ultrasonic sensor more accurately than commonly used single-lobe Gaussian approximations [1, 5, 15, 16, 18].

1.1 Self-Referencing in Proper Perspective

Confirming position and orientation using sensors presupposes that the employed sensors can produce predictable and reliable range readings and that references are accurately mapped and frequently within range. Provided the sensors and objects are configured accordingly self-referencing should be a fairly simple task of comparing the range readings and/or feature signatures to what is expected based on where the robot thinks it is [16, 17]. In truth, though, real sensors are not always reliable and environments are not always accurate nor static.

As unrealistic as it is for a global motion planner to assume that the robot will be able to traverse along the planned trajectory free of unexpected events, so too is it unrealistic to assume that self-referencers will always get accurate information. Sensor noise is a big enough problem in and of itself, despite the availability of various filtering strategies [2, 3, 17]. But the mere presence of moving robots and people and the likelihood that furniture, equipment, etc. will be rearranged can significantly obscure the image a robot might get of its surroundings. Realistically, the potential exists that a mere 10-20% of the range readings would accurately reflect the expected surrounding geometric beacons. Further, behaviors like unexpected-obstacle avoidance and mapping might be prioritized over self-referencing in certain settings. Hence, if a mobile robot is to function in a realistic, dynamic environment, it needs to have more than just the high-level global motion planning and the mid-level self-referencing. It must also have the ability to filter out extraneous readings (self-localization), regardless of how dense they are, and do low-level obstacle avoidance when necessary [19].

1.2 Approach

There are three fundamental steps necessary for solving the self-referencing problem: how to represent the objects in the environment, how to search for geometric beacons based on this representation and how to predict a reliable range reading. In [9, 10] we proposed that, whenever possible, objects be represented geometrically on the basis of surface information quality and compatibility with global motion planning [8]. In [13, 14] it was shown that t-vectors, in addition to their efficient collision detection, can quickly identify front and rear surfaces of a polygon relative to a point (e.g., a transducer). This paper will do the following: First, it will discuss how t-vectors and C-space improve the efficiency of collecting geometric beacons inside a *sensor window* and, hence, in range of an actual sensor. Second, it will present a neural network based approach that computes a probabilistic credence factor for all possible predictions of geometric range and angle relative to cone normal. Third, it will propose a strategy for predicting a sensor's most likely range, reflective angle and angle relative to cone normal. Collectively, these processes can provide a filter or control strategy with a stochastic certainty of each range prediction.

2 Sensor Boundaries and Detectability

Knowing the boundaries of a theoretical cone model (maximum and minimum ranges and peripheral limits) enable the cone to be reconstructed and used as a moving *window* anywhere on the map to correspond to the robot's position and orientation. Objects inside this window can be *seen* or *heard* by the robot. Hence, if the robot's position and orientation are known, and the *sensor window's* position and orientation relative to the robot are known, any object on the map inside the window's boundaries could be used as a geometric beacon. The sensor window boundaries in this section are modeled after the ultrasonic sensor.

2.1 Sonar Boundaries and Descriptors

Due to their radiative properties, ultrasonic sensors create conical regions of propagating sound waves. If an object is inside this conical region, it can produce an echo. Figure 1 shows crisp boundaries for an ultrasonic sensor. The cone half-width, θ_o, defines the angle beyond which an echo is no longer expected to be received. For pulse-type ultrasonic sensors, the minimum range, $R_{\min}$, also called the near or Fresnel zone, is the distance from the transducer to the end of the near region [1]. For continuous-wave ultrasonic sensors, $R_{\min}$ is set by the *ringdown* [4]. The maximum range, $R_{\max}$, for a pulse-type sonar depends on where in the sonar cone an echo is produced and whether or not the sensor is supported by a time controlled gain amplifier [20]. It must be emphasized that boundary descriptors depend on the type of sensor being used.

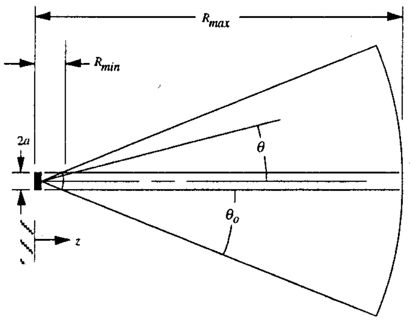


Figure 1. Sonar cone boundary descriptors.

2.2 Detectability Factors

There are numerous factors that influence actual object detectability. That is, an object might be within range of the sensor and still not produce an echo that returns to its source. Some reasons for why an in-range *geometric beacon* [17] might not be detected include: First, the sonic pressure might not be strong enough at the echo point. Second, the surface dimension, d_s, might be smaller than the sonar's wavelength ($d_s < \lambda$), resulting in a weak echo pressure. (Hence, the smallest surface dimension, an object in the environment can have to be detectable is $d_{S_{\min}} \geq \lambda$.) A third possibility is that the nearest reflecting surface is at such a reflective angle, δ , to the cone arc that the echo could be too weak to register with the sensor. Although it varies with surface texture, a maximum cone arc deviation, $\delta_{\max}$, can be experimentally determined. It will be shown later that $\delta_{\max}$ helps establish the criteria used to detect the smallest assumed geometric beacons in the environment.

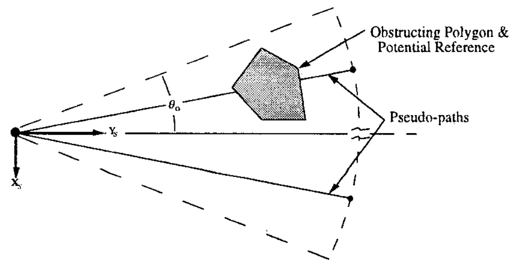


Figure 2. Pseudo-paths detecting potential geometric beacon.

3 The Sonar Cone Window Model

To *detect* objects in a scan cone, a set of *pseudo-paths* radiating from the transducer are tested against polygons in the environment for collision [9]. Collision detection is most efficient with the t-vector test presented in [13, 14]. The pseudo-path premise is: if a pseudo-path inside the boundaries of the sonar window is obstructed by a polygon, then an actual object that can produce an echo is, according to memory and estimated position and orientation, within sonar range. If the pseudo-paths are spaced such that even the smallest objects in the environment can be detected, then *every* mapped object can be used as a reference.

The start point of each pseudo-path is the origin of the sonar frame. The end point locations, $G(x,y)_S$, in the sonar frame depend on θ_o, $R_{\max}$, $d_{S_{\min}}$, $\delta_{\max}$ and if the C-space buffer (polygons grown by thickness t_b to keep robot motion a safe distance from objects) is used. Specifically, the end points of the pseudo-paths should be a distance greater than or equal to $R_{\max}$ from the transducer surface (origin) and not separated by more than $w_{\min}$, the worst-case projection of $d_{S_{\min}}$ at $\delta_{\max}$. Figure 3 shows $d_{S_{\min}} > w_{\min}$.

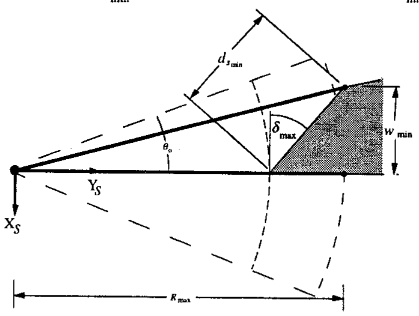


Figure 3. Pseudo-path separation not to exceed $w_{\min}$.

So pseudo-paths have a maximum allowable separation of

$$w_{\min} = d_{s_{\min}} \cos(\delta_{\max}). \tag{1}$$

Knowing θ_o, $R_{\max}$ and $w_{\min}$, the number of pseudo-paths

$$N_{pp} = \left\lceil \frac{(2)(R_{\max})(\tan\theta_o)}{\text{Wmin}} \right\rceil + I_b \tag{2}$$

where I_b depends on if the polygons have C-space buffers.

$$I_b = \begin{cases} +1 & (tb = 0) \\ -1 & (tb > 0) \end{cases} \tag{3}$$

So, the actual pseudo-path separation, w, is

$$w = \left\lceil \frac{(2)(R_{\max})(\tan \theta_o)}{(N_{PP} - I_b)} \right\rceil \leq w_{\min} \tag{4}$$

Pseudo-paths start at $S(x,y)_s = (0,0)_s$ and end at

$$G_N(x,y)_s = \{[(R_{\max})(\tan \theta_o) - (N - J_b)(w)], R_{\max}\}_s$$
$$\text{for } (N = 1,2,\ldots N_{pp} + J_b) \tag{5}$$

J_b also depends on if the polygons have C-space buffers.

$$J_b = \begin{cases} 1 & (t_b = 0) \\ 0 & (t_b > 0) \end{cases} \tag{6}$$

Figure 4 shows the resulting sensor window model.

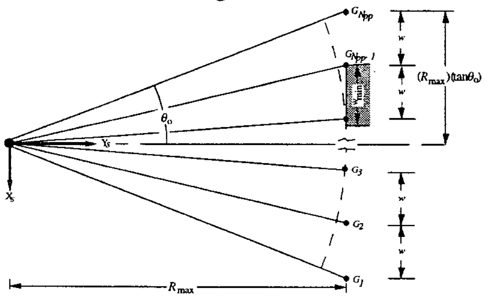


Figure 4. Pseudo-paths for a scan cone window.

Proposition 1: *The C-space buffer, t_b, increases the detectability of geometric beacons and, hence, reduces the number of pseudo-paths, N_{pp}, for a sonar cone window.*

Proof: The virtues of C-space buffers can be demonstrated by comparing two examples. The first will be where there are no buffers; the second will be with buffers.

Example 1: A surface can have a dimension of $w_{\min} = 2$ cm and still be detectable by most ultrasonic sensors. So if the sonar cone has values of $R_{\max} = 240$ cm, $\theta_o = 35^{\circ}$, $\delta_{\max} = 15^{\circ}$ and $t_b = 0$, then $N_{pp} = 174$. That is, 174 pseudo-paths will be required for a two dimensional cone window if there are no buffers and the smallest expected dimension is 2 cm.

Clearly *this is too many pseudo-paths* to inspect each referencing cycle. A better solution can be found if polygons are buffered (enlarged by t_b) because the polygon's likelihood of being detected is substantially increased.

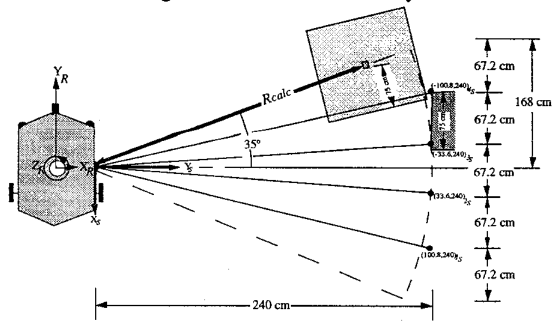


Figure 5. Pseudo-paths for geometric beacon with C-space buffer.

Example 2: Keeping $R_{\max} = 240$ cm and $\theta_o = 35^{\circ}$, and letting $d_{s_{\min}} = t_b = 75$ cm we reduce N_{pp} to four. Herein lies the advantage of C-space buffered polygons. See figure 5.[]

4 Range Credence Estimation

It was noted in [9, 10] that sonar readings can not be based entirely on a crisp geometric model of the sensor's boundaries. That is, a surface's detectability credence κ is a continuous function of range, cone radial angle, cone arc deviation and myriad other variables that cannot, in general, be decoupled in independent thresholds.

$$\kappa = f\left(R, \theta, \delta, T_{amb}, d_{s_{\min}}, \ldots\right) \qquad 0 \leq \kappa \leq 1 \tag{7}$$

One can approximate a credence factor based on range and radial angle from the Gaussian curve that is commonly used to model pressure amplitude [15]. But the single-lobe Gaussian approximation is not truly representative of actual ultrasonic sensor behavior. The actual pressure curve can be more accurately modeled by a multi-layered neural network. Multi-layered neural nets are known for their ability to model continuous output functions from multi-dimensional input problem spaces [7, 21, 23]. In this section we present a credence factor model based on two of the variables in (7): range R and radial angle θ.

4.1 Neural Network Architecture and Training

While there exists a variety of feasible configurations, the architecture we used is the single-hidden-layer, fully connected feed-forward neural net shown in Figure 6. To enhance back propagation, adaptive learning rates [21] and normalized weights were used to train the neural net.

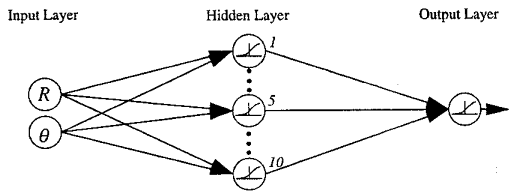


Figure 6. Multi-layer neural network used to learn range credence.

4.2 Training and Recall Data

Training and testing data was collected from a long reflective surface (wall) using a Polaroid transducer. Shown in figure 7, a 399-node grid was used to model half of the symmetric cone ($R_{increment} = 250\text{mm}$ and $\theta_{increment} = 2.5^{\circ}$). Room temperature was kept at 23°C. Distance was measured along the transducer normal. For the pulse sonars on MARGE (Mobile Autonomous Robot for Guidance Experiments) $R_{\min} = 500\text{mm}$ and $R_{\max} = 5500\text{mm}$. The unaltered Polaroid sonar is usually modeled with $\theta_o = 30^{\circ}$. However, to account for all possible lobes, the range of angles was $\theta \in \left[-45^{\circ}, 45^{\circ}\right]$ in 2.5° increments. Five training and testing data sets comprised of 100 readings at each of the 399 grid nodes were collected on different days.

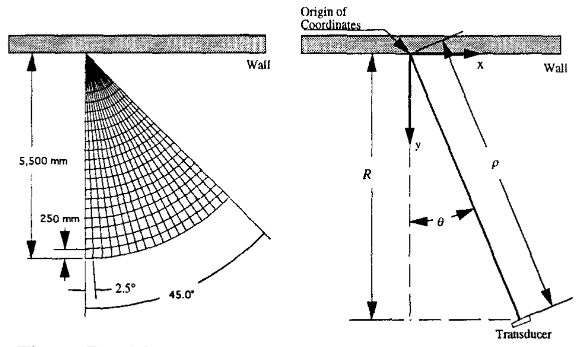


Figure 7. (a) Node grid of half-cone and (b) reference frame.

The credence value for each example was assumed to be the fraction of correct readings over the total number of readings for each location. Figure 8 shows the observed credence surface in two reference frames: the horizontal axes of figure 8a are in units of range (R mm) and radial angle ($\theta°$); the horizontal axes of figure 8b are (X_s, Y_s) coordinates in the sonar cone. Both frames use the vertical axis to measure stochastic confidence $\kappa \in [0, 1]$. Of particular interest is that, unlike the single-lobe Gaussian approximations, the actual data in figure 8 suggests that all *three* lobes be considered when a robot is self-referencing.

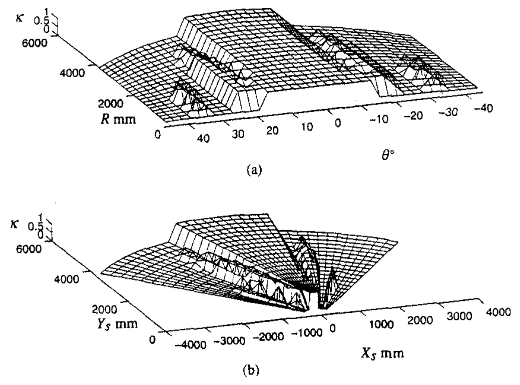


Figure 8 Observed κ-surface in: (a) R-θ and (b) (X_s, Y_s).

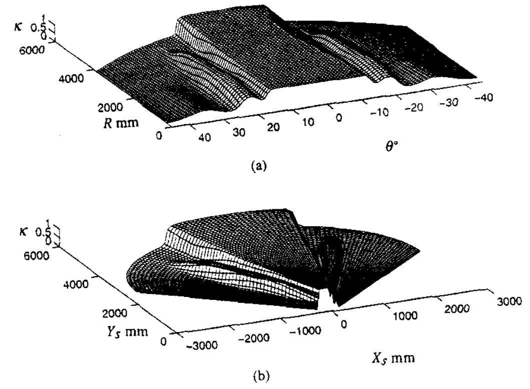


Figure 9. Modeled κ-surface in: (a) R-θ and (b) (X_s, Y_s).

4.3 Modeling Sonar Credence with a Neural Network

Figure 9 shows the neural net's mapping of the data in figure 8. In general, the 2-10-1 neural net accurately and smoothly modeled the credence of predicted range readings over the entire three-lobed spectrum of R-θ combinations. Recall is extremely fast because the neural net accepts raw R-θ data and the overall size of the architecture is small.

5 Range Calculations

Range calculations to geometric beacons are more complicated than simply computing the distance from the transducer to the closest *feature*. Other attributes must also be factored in. While the feature identification model in [9] helps make geometric maps compatible with self-localization approaches like multiple hypothesis tracking (MHT) [3], it tends to misrepresent geometric beacons. We propose that, instead of relying on generic feature labels, at least the following four geometric- and neural net-based tests help consider how reliably a surface returns an echo. The tests check: 1) if the surface is visible to the transducer; 2) if the surface's cone arc deviation is within tolerable limits; 3) if the candidate calculated range is within credible limits; and 4) if the candidate echo point is occluded by another in-range polygon.

Generality and complexity should be considered for algorithms based on the myriad possible signature test sequences and combinations used to predict a sensor range and confidence level. Some tests might be used in parallel and others in sequence; some might require more in-depth processing. Further, there are several other tests that can complement the signature tests. For example, the cone arc deviation might be included as a third variable for training the neural network mentioned in section 4. Finally, the signature tests can be crisp and/or fuzzy [22]. Either way, it should be noted that computing a predicted range and corresponding credence factor is an iterative and, in many ways, heuristic process.

5.1 Signature Test 1: Surface Visibility

A *rear* surface on a geometric beacon will never return a direct echo. Only portions of an object that are visible to a sensor will produce a signature. How to efficiently identify which surfaces are *front* (i.e., visible) and which are *rear* using t-vectors is presented in [13, 14]. Hence, if the surface in question is a *front* surface, it is safe to move to the second test since this type of surface can return an echo.

5.2 Signature Test 2: Cone Arc Deviation

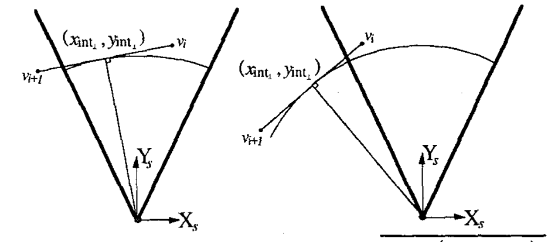
If the visible surface $\overline{v_i v_{i+1}}$ is inside the cone enough to potentially send an echo to the transducer, we calculate a candidate predicted range, $R_{calc_{candidate}}$, and examine the cone arc deviation, δ of the surface at that range. In this section we consider three cases of surface configurations relative to the sensor window. Case I considers the possibility that a surface has no cone arc deviation $(\delta = 0^\circ)$. This must be checked for first because a surface's geometrically closest point is where a radial from the transducer is perpendicular to that surface. Of course, being geometrically closest does not imply a high enough confidence level, $\kappa < \tau_\kappa$. Hence, if the R- θ inputs that correspond to a surface with $\delta = 0^\circ$ does not yield a credence factor value above an established threshold, $\kappa \geq \tau_\kappa$, then testing should resume at Case II (surface is completely inside the cone window)or Case III (surface straddles or is partially inside the cone window).

5.2.1 Case I: Cone Arc Deviation Equals Zero

We recall that the shortest distance between a point x and a line segment $\overline{v_i v_{i+1}}$ is the distance along a line through x and perpendicular to $\overline{v_i v_{i+1}}$. For $\delta = 0^\circ$, this implies that a line perpendicular to $\overline{v_i v_{i+1}}$ passes through the origin of the sensor reference frame. See figure 10. If the intersection point $(x_{\text{int}_\perp}, y_{\text{int}_\perp})_s$ is either outside the sensor window **or** not on the surface $\overline{v_i v_{i+1}}$, then we move to Case II. If, however, $(x_{\text{int}_\perp}, y_{\text{int}_\perp})_s$ is inside the sensor window **and** on the surface $\overline{v_i v_{i+1}}$, we can calculate the range by,

$$R_{calc_{candidate}} = \sqrt{x_{\text{int}_\perp}^2 + y_{\text{int}_\perp}^2} \tag{8}$$

If $R_{calc_{candidate}}$ and θ do not satisfy $\kappa \geq \tau_\kappa$, we go to Case II.



$(x_{\text{int}_\perp}, y_{\text{int}_\perp})$
v_i
v_{i+1}
Y_s
X_s
$(x_{\text{int}_\perp}, y_{\text{int}_\perp})$
v_i
v_{i+1}
Y_s
X_s

Figure 10. For $\delta = 0$, line $\perp$ to $\overline{v_i v_{i+1}}$ is $\overline{(0,0)_s (x_{\text{int}_\perp}, y_{\text{int}_\perp})_s}$.

5.2.2 Case II: Surface Completely Inside Sensor Window

For surfaces completely inside the sensor window and not perpendicular to the sensor origin, we first calculate the range to each vertex and then determine which portion of the surface will yield the smallest arc angle deviation.

Proposition 2: *For a surface $\overline{v_i v_{i+1}}$ that is completely inside the sensor window, the smallest arc angle deviation, δ, is at the closest vertex (either v_i or v_{i+1}).*

Proof: This theorem can be proven graphically. Figure 11 shows a line intersecting three concentric semi-circles centered on the same origin. The larger the circle, the greater its sonar range. It can be seen that $\overline{v_i v_{i+1}}$ is tangent only to the smallest circle. That is, the point of intersection (and tangency) between the line and circle of smallest radius, R_0, is closest to the origin. Likewise, because the line is tangent, $\delta_0 = 0^\circ$. Intersections between the line and outer two semi-circles indicate that as range increases so does arc angle deviation. Hence, the closest vertex produces the smallest δ.[]

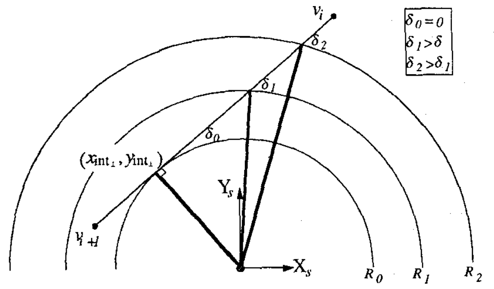


Figure 11. As range increases, so does the arc angle deviation.

The two ranges for the surface endpoints are,

$$R_{v_i} = \sqrt{x_{v_i}^2 + y_{v_i}^2} \text{ and } R_{v_{i+1}} = \sqrt{x_{v_{i+1}}^2 + y_{v_{i+1}}^2} \tag{9a, b}$$

For the surface, $\overline{v_i v_{i+1}}$, the angle $\varphi_{v_i v_{i+1}}$ can be found from,

$$\varphi_{v_i v_{i+1}} = \tan^{-1}\left(\frac{y_{v_i} - y_{v_{i+1}}}{x_{v_i} - x_{v_{i+1}}}\right) \tag{10}$$

and the cone arc angles at each vertex can be found by,

$$\varphi_{v_i} = \tan^{-1}\left(\frac{y_{v_i}}{x_{v_i}}\right) \text{ and } \varphi_{v_{i+1}} = \tan^{-1}\left(\frac{y_{v_{i+1}}}{x_{v_{i+1}}}\right) \tag{11a, b}$$

The arc angle deviation at each vertex is, thus,

$$\delta_{v_i} = \left\| \varphi_{v_i v_{i+1}} \right| - \left| \varphi_{v_i} \right\| \text{ and } \delta_{v_{i+1}} = \left\| \varphi_{\overline{v_i v_{i+1}}} \right| - \left| \varphi_{\overline{v_{i+1}}} \right\| \tag{12a, b}$$

5.2.3 Case III: Surface Straddles or is Partially In Window

Having one or both vertices outside the cone forces us to clip the surface at the cone boundaries. See figure 12. Substituting the clip vertices for their respective surface endpoints permits us to use the same procedure in Case II.

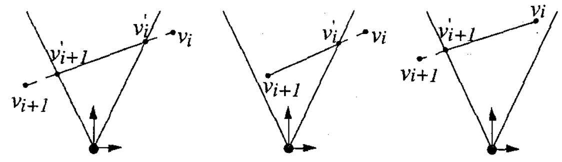


Figure 12. Clipping portions of surface not in sensor cone window.

A crisp approach to this problem might require that if equation 12 is less than a maximum allowable arc angle deviation, $\delta_{\max}$, the surface could remain a reference candidate. If, on the other hand, the smallest δ achievable from surface $\overline{v_i v_{i+1}}$ is larger than $\delta_{\max}$, the surface could be considered an unreliable reference. After all, it was proven in Proposition 2 that the further away from the closest vertex one travels, the higher the value for cone arc deviation δ. An even better solution might be to include the δ as a variable in the range credence calculation.

5.3 Signature Test 3: Credible Calculated Range

Now we must determine if the credence factor $\kappa = f\left(R_{calc_{candidate}}, \theta, \ldots\right)$ is high enough to expect that the actual range reading will correspond to the predicted range. If κ is less than some minimum credence value, $\kappa_{\min}$ then the $\overline{v_i v_{i+1}}$ should be examined in greater detail. That is, one should not simply assume that a surface is altogether unreliable if its nearest expected echo point yields a credence factor $\kappa < \kappa_{\min}$; the nearest point on $\overline{v_i v_{i+1}}$ might actually be in a ravine (between lobes) on the κ-surface. To examine a surface in greater detail, one could segment the $\overline{v_i v_{i+1}}$ in increments between the two vertices. Then, moving from the nearest vertex to the farthest vertex, compute a new $R_{calc_{candidate}}$, θ, κ and δ at each increment until the most reliable candidate range calculation is found. One could also compute a weighted sum to predict the bilateral tolerance on the expected value of $R_{calc_{candidate}}$. Figure 13 shows $\overline{v_i v_{i+1}}$ decomposed into subsurfaces and a plot of κ versus the distance from v_i to v_{i+1}.

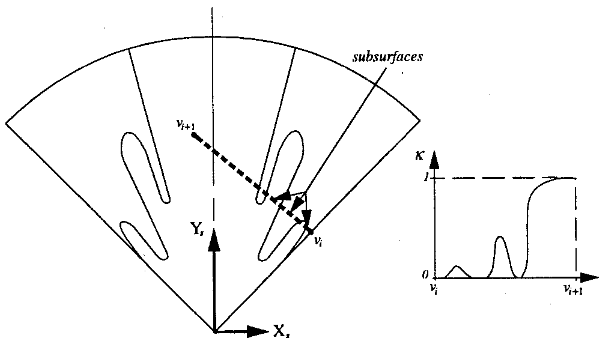


Figure 13. Decomposed $\overline{v_i v_{i+1}}$ with credence κ per subsurface.

5.4 Signature Test 4: Obstructed Range Point

Since it is possible for several geometric beacons to be in a sensor window, the point to which $R_{calc_{candidate}}$ was calculated must be checked for occlusion by other in-range polygons. This is easily done using t-vectors to detect collisions between the other in-range polygons and a line connecting the transducer and echo point on $\overline{v_i v_{i+1}}$. If a collision is detected, the candidate calculated range is assumed unreliable because no pulse will be able to directly reach the candidate echo point . If, on the other hand, the line is collision free, it can be assumed that a sonar pulse will reach the echo point and, hence, return a valid range.

5.5 General Comments

Searching the polygon map in this manner fills the window model with all mapped geometric beacons in range of a given sensor. The contents of this window are then examined to calculate the range, R_{calc} to the closest reliable feature. R_{calc} is soon after compared with the observed range reading, R_{obs}, to either i) confirm the robot's position and orientation, ii) detect a previously unknown obstacle, iii) find that an obstacle was removed, or iv) cause the robot to change its dead reckoning values of position and orientation. There are several techniques with which the aforementioned self-referencing functions can be performed. The crudest method involves setting a threshold on the difference between R_{obs} and R_{calc} and immediately changing the dead reckoning values when the difference exceeds a threshold. Smoother methods can be found in line fitting [6] or the more refined Extended Kalman Filtering [17].

These (and other) signature tests were confirmed against the environment based on actual robot position, orientation and sonar readings. MARGE was used to justify the use of sensor windows and the applicability of t-vectors, C-space-time, geometric representation and the neural net credence model for self-referencing. It was assumed that since the calculated sonar ranges accurately reflected MARGE's sonars, the model prescribed here is valid.

References

[1] B. Barshan and R. Kuc, "Differentiating Sonar Reflections from Corners and Planes by Employing an Intelligent Sensor." *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 12 no. 6, pp. 560-569, June 1990.
[2] C. Brown, H. Durrant-Whyte, et al., "Centralized and Decentralized Kalman Filter Techniques for Tracking, Navigation and Control." in *Proc. DARPA Image Understanding Workshop*, 1989.
[3] I. J. Cox and J. J. Leonard, "Probabilistic Data Association for Dynamic World Modeling: A Multiple Hypothesis Approach." in *Fifth International Conference on Advanced Robotics*, pp. 1287-1294, 1991.
[4] 4ermotion, User's Manual. 5457 Aerospace Road; Roanoke, VA.
[5] A. Elfes, "Sonar-Based Real-World Mapping and Navigation." *Journal of Robotics and Automation* Vol. 3 no. 3, pp. 249-265, June 1987.
[6] H. R. Everett, G. A. Gilbreath, et al. (1990). Modeling the Environment of a Mobile Security Robot. *Tech. Doc. 1835*. San Diego, CA, Naval Ocean Sys Ctr.
[7] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Company, New York, 1994.
[8] Y. K. Hwang and N. Ahuja, "Gross Motion Planning-A Survey." *ACM Computing Surveys* Vol. 24 no. 3, pp. 219-291, September 1992.
[9] J. A. Janét, R. C. Luo, et al., "Sonar Windows and Geometrically Represented Objects for Mobile Robot Self-Referencing." *IEEE/RSJ International Conference on Intelligent Robotics and Systems*, 1993.
[10] J. A. Janét. Global Motion Planning and Self-Referencing for Autonomous Mobile Robots. North Carolina State University, Raleigh, NC, 1994.
[11] J. A. Janét, S. G. Goodridge, Ren C. Luo, "Dynamic Motion Control of Sensor-Based Autonomous Mobile Robot." *APS International Conference on Mechatronics and Robotics*, Aachen, Germany, April 1994.
[12] J. A. Janét, R. C. Luo, M. G. Kay, "Autonomous Mobile Robot Motion Planning Enhanced with Extended Configuration-Space and Half-Planes." *IEEE/SICE/AEI Int'l Conf.on Industrial Electronics*, Bologna, Italy, Sept. 1994.
[13] J. A. Janét, R. C. Luo, M. G. Kay, "Traversability Vectors Make Autonomous Mobile Robot Motion Planning and Self-Referencing More Efficient." *IEEE/RSJ Int'l Conf. on Intelligent Robotics and Systems*, Sept. 1994.
[14] J. A. Janét, R. C. Luo, M. G. Kay, "Autonomous Mobile Robot Global Motion Planning and Geometric Beacon Collection Using Traversability Vectors." *accepted to IEEE Trans. on Robotics and Automation*, 1995.
[15] R. Kuc, "A Spatial Sampling Criterion for Sonar Obstacle Detection." *IEEE Trans. on Pattern Analysis and Machine Intelligence* Vol. 12 (7), July 1990.
[16] R. Kuc and V. Viard, "A Physically Based Navigation Strategy for Sonar-Guided Vehicles." *Int. J.of Robotics Research* Vol. 10(2), April 1991.
[17] [J. Leonard and H. Durrant-Whyte, *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers. Norwell, Massachusetts. 1992.
[18] H. Moravec and A. Elfes, "High Resolution Maps From Wide Angle Sonar." in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 116-121, 1986.
[19] F. Noreils, "Toward A Robot Architecture Integrating Cooperation Between Mobile Robots: Application to Indoor Environment." *International Journal of Robotics Research* Vol. 12 no. 1, pp. 79-98, Feb 1993.
[20] Polaroid, *Ultrasonic Ranging System*. Cambridge, MA, 1982.
[21] M. Smith, *Neural Networks for Statistical Modeling*. Van Nostrand Reinhold, New York, NY, 1993.
[22] T. Terano, K. Asai, M. Sugeno, Fuzzy Systems Theory and Its Applications, Academic Press, 1992.
[23] J. M. Zurada, *Introduction to Artificial Neural Systems*. West Publishing Company, St. Paul, MN, 1992.