

Environmental Mapping Using Distance Ranging for Self-Aware Locomotion

EE202A Final Project
Dan Lander, Huiyu Luo

Abstract

In this project, my partner and I are trying to develop a self-aware locomotive range sensing device that detects the presence of static objects near and around embedded computing sensor nodes. This object range detection is done by an ultra sound sensor. The range sensor is planned to be connected to a LINUX hardware/software platform via the mica2 microcontroller developed at UC Berkeley. The platform will be tied to a cable and able to move in one direction. This enables the range sensor to survey its surrounding environment by determining the distances to nearby objects. These distances are then translated into projected coordinates in the form of a propagation matrix by an object-modeling algorithm. Once all static object coordinates are processed and mapped out by this algorithm, distinct obstructing objects in view can be located. Once determined, a mobility control algorithm can then be suggested and formulated to accurately relocate the corresponding obstructed sensor node to nearby unobstructed positions.

Introduction

Image processing has been considered as one of the choices to direct the self-aware locomotion of embedded sensor nodes. The sensor node captures and processes as much information as it can on the imagery of the object it is confronted with. Unfortunately, the sensor node cannot learn a whole lot about the positioning of the object strictly by acquiring knowledge of the object's imagery. Hence, it is unlikely that the sensor node will be able to accurately move towards an unobstructed position to continue optimally surveying the surrounding environment. Other techniques need to be explored in order for self-aware locomotion to be more successful. That is where environment mapping using distance ranging comes into play and why it is needed.

Our approach to environmentally mapping out obstructing objects using distance ranging involves the use of an acoustical range sensor. This device coordinately projects where objects might be found in view of the sensor using an ultra sound beam. This approach also uses data processing mechanisms such as object mapping for determining exactly where the static objects may be situated. There is also a suggested mobility control algorithm for determining a possible new set of coordinates for the sensor node to relocate in order to avoid obstruction. This approach based on environmental range sensing is shown here to be promising.

In what follows ahead in this paper, a more comprehensive description of how we intend to approach this challenge of creating self-aware locomotion through distance

ranging will first be given. Next, a complete explanation of our methodology used in this project will be described in further detail. This will be followed by a discussion and interpretation of our results. Finally, the paper finishes with major conclusions drawn from our findings and an evaluation of how successful we thought we were in handling our challenge of using distance ranging to map out potentially obstructing objects.

Background/Theory to our Algorithms

I. Object mapping algorithm

Our approach to distance ranging will involve two major algorithms. The first, described in this section, is an object mapping algorithm. The algorithm will first extract all recorded distances from the surrounding environment surveyed by the range sensor and use that data to construct a propagation matrix. The entries to the propagation matrix will involve a position and a direction denoting the measured distance sampled as shown below.

$$P = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1J} \\ P_{21} & P_{22} & \cdots & P_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ P_{I1} & P_{I2} & \cdots & P_{IJ} \end{bmatrix}$$

In this equation, entries P_{ij} are the distance from certain observing position and angle. Row i denotes the angular direction θ and column j denotes the position n of the sensor node. The position indicates where the range sensor was located and the direction indicates at what angle relative to the sensor the measured distance was taken at. An example displaying this is shown in the figure below.

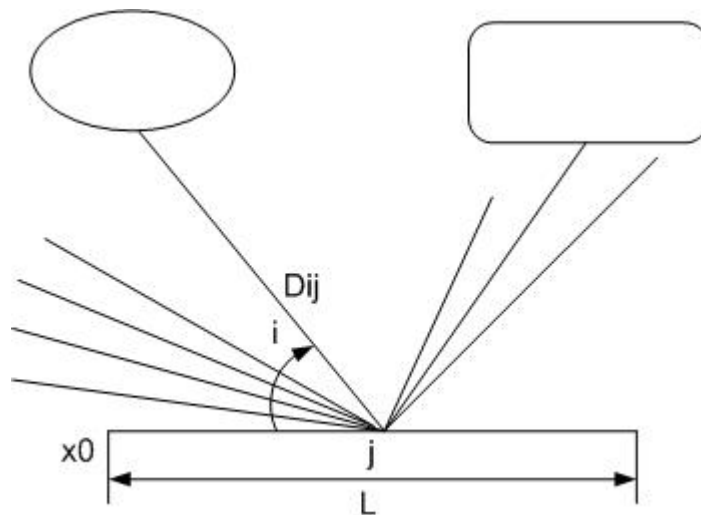


Figure 1. A sensor field with obstacles and the range sensor

The range sensor beamwidth is around 15 degrees, so the closest objects within that width will be reported. The sensor view range limit is about 10 meters, so any objects beyond this distance will not be considered. What this means is that it can safely be assumed that no objects exist within view of the sensor sampled at position n in direction θ if the distance measured there is 10 meters. This will correspond to an entry of 10 in the propagation matrix at row n and column θ . However, if an entry of less than 10 is recorded in the propagation matrix at some row n and column θ , it can then be assumed that an object does lie in view of the range sensor situated at position n in direction θ . This is how the distance entries in the propagation matrix can be interpreted to determine whether an object lies in view of some position n and direction θ .

Once the propagation matrix is formulated, the distance entries will be used as input by the mapping algorithm to determine what the corresponding (x, y) coordinates are for those entries in the matrix that have distances less than ten. This should produce a set of coordinates projecting where possible obstructing objects may lie. Since this set may produce duplicate coordinates due to the fact that two or more distinct locations may be viewing the same object position, the algorithm will only produce a set of distinct coordinates of where an object may be situated. Using this set of coordinates, individual objects can be mapped and located for further use of telling a range sensor where to relocate to if obstructed by one of these objects. This is done by a mobility control algorithm which is described in more detail later in the next section. Once all possible object coordinates have been determined, the mapping algorithm will finally attempt to produce a graphical display of possible object boundaries at each sensor point n taken along the horizontal viewing line. Each graphical display taken will consist of a distance d along the y -axis and an angle θ along the x -axis. Hence, for each direction θ sampled by the sensor at some location n , the measured distance will be plotted along the y -axis yielding a rough sketch of a possible object boundary if the sensor's view were situated at position n . Each of these graphs serves to give more information about what the sensor is actually seeing and how the object boundaries change with sensor position.

For the remainder of this section, the theoretical details behind how this mapping algorithm determines the corresponding coordinates from the initial propagation matrix entries will be discussed and outlined in detail. To begin with, given a measured distance d less than ten in some row n and column θ of the propagation matrix, the object mapping algorithm computes the corresponding coordinates of the possible object (x, y) using coordinate geometry. This involves using the distance formula and some trigonometry. Since we have two unknowns x and y , we will need to formulate two equations to solve for each unknown. Assume that (dx, dy) are the known coordinates of the range sensor. These two equations can be formulated as follows:

- a.) $\tan(\theta) = (y-dy)/(x-dx)$ where \tan is the tangent function and x, y are the coordinates of the object we are seeking to determine (our two unknowns).
- b.) $d = ((x-dx)^2 + (y-dy)^2)^{0.5}$ with $d < 10$ and x, y being the coordinates of the object we are seeking to determine (our two unknowns).

Now since our range sensor can only move horizontally in one direction along the x-axis at discrete points of n each spaced 0.5 meters apart, this allows for some simplifying conditions. The coordinates of the range sensor are now $(dx=0.5n, dy=0)$. This simplifies the above two formulas which may now be expressed as follows:

- c.) $\tan(\theta) = y / (x-0.5n)$ with $dy=0$ and $dx=0.5n$
- d.) $d = ((x-0.5n)^2 + y^2)^{0.5}$ with $d < 10$, $dy=0$ and $dx=0.5n$

Since we have θ , n and d from our propagation matrix entry, we can now rearrange one of the equations and substitute it into the other to solve for x and y respectively. This yields equations for x and y as shown below:

- e.) $x = (d^2 - y^2)^{0.5} + 0.5n$
- f.) $y = (\tan(\theta))(x-0.5n) = (\tan(\theta))((d^2-y^2)^{0.5} + 0.5n - 0.5n)$ after substituting in for x in the y expression above.

Now we rearrange the expression in (f) to get y on one side of the equation. This will yield a final expression for y as follows:

- i. $y = (\tan(\theta))((d^2-y^2)^{0.5})$ after canceling out $0.5n$'s
- ii. $y^2 = (\tan(\theta))^2 (d^2-y^2)$ after squaring both sides
- iii. $y^2 = d^2(\tan(\theta))^2 - y^2(\tan(\theta))^2$ after distributing the right side
- iv. $y^2 + y^2(\tan(\theta))^2 = d^2(\tan(\theta))^2$ after rearranging both sides
- v. $y^2(1 + (\tan(\theta))^2) = d^2(\tan(\theta))^2$ after factoring y^2
- vi. $y^2 = d^2(\tan(\theta))^2 / (1+(\tan(\theta))^2)$ after dividing by $1+(\tan(\theta))^2$
- vii. $y^2 = d^2(\tan(\theta))^2 / (\sec(\theta))^2$ after applying a trig identity
- viii. $y = d(\tan(\theta)) / (\sec(\theta))$ after square rooting both sides
- ix. $y = d(\tan(\theta))(\cos(\theta))$ after reciprocating the secant function
- x. $y = d(\sin(\theta)/\cos(\theta))(\cos(\theta))$ after applying a trig identity to tangent
- xi. $y = d(\sin(\theta))$ after canceling out cosine functions

Now substituting the last y expression in (xi) back into the x expression written in (e) will yield a final expression for x as follows:

- i. $x = (d^2 - (d(\sin(\theta)))^2)^{0.5} + 0.5n$ after substituting in $y=d(\sin(\theta))$
- ii. $x = (d^2 - d^2(\sin(\theta))^2)^{0.5} + 0.5n$ after squaring $d(\sin(\theta))$
- iii. $x = (d^2(1 - (\sin(\theta))^2))^{0.5} + 0.5n$ after factoring out d^2
- iv. $x = d(1 - (\sin(\theta))^2)^{0.5} + 0.5n$ after square rooting d^2
- v. $x = d(\cos(\theta))^2)^{0.5} + 0.5n$ after applying a trig identity
- vi. $x = d(\cos(\theta)) + 0.5n$ after square rooting the cosine function

Hence, the important final two expressions for x and y yielding our desired object coordinates (x, y) are:

- g.) $x = d\cos(\theta) + 0.5n$ with $d < 10$ and n , θ given from the matrix entry
- h.) $y = d\sin(\theta)$ with $d < 10$ and θ given from the matrix entry

This concludes the theoretical derivation of how this algorithm will compute the projected object coordinates given the corresponding initial distance entries at various points n and directions θ in the propagation matrix.

II. Mobility control algorithm

The second of our major algorithms, described in this section, is a mobility control algorithm. This algorithm will first take in a specific target coordinate (x, y) to determine whether that target position is initially visible from the present sensor location. Note, we are assuming that the target coordinates lie within the 10 meter visibility range of the sensor and that the target is situated along a direction θ that was originally sampled by the sensor. If the target is visible by the range sensor, the predetermined distance to the target computed from (x, y) should be less than or equal to the measured distance surveyed by the range sensor. If the target is not visible, then the actual surveyed distance should be less than the predetermined distance indicating that an object exists in front of the desired target. The details of how this algorithm exactly accomplishes these steps are further described and explained below.

The predetermined distance to the target point relative to the range sensor's current position is computed using the Pythagorean Theorem. The algorithm first determines the net difference $(\Delta x, \Delta y)$ from the range sensor to the target point and then uses that as input into the Pythagorean Theorem to compute the predetermined distance. The formulas are as follows:

- a.) $(\Delta x, \Delta y) = (x_2 - x_1, y_2 - y_1)$ with (x_1, y_1) the coordinates of the range sensor and (x_2, y_2) the coordinates of the target.
- b.) $\text{predetermined distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Once this predetermined distance is calculated as shown above, the next step is to compare this distance to the one already measured and recorded as an entry in the propagation matrix from the first algorithm. If the predetermined distance is less than or equal to the distance already measured, then the target is viewable. However, if the measured distance is less than the predetermined distance, then the target is not viewable and there is likely an obstructing object in front. To determine which distance entry in the matrix to use as the measured distance in the comparison, we need to know where the sensor is situated and at what angular direction relative to the sensor the target point is located. This requires finding the sensor's position point n and the angle θ the sensor makes with the target location. Once we have these two values, we can go look up the entry in the matrix that corresponds to row n and column θ and use that measured distance to compare with our predetermined distance. These two values n and θ are determined as follows:

- c.) $n = d / 0.5$ where d is the current x coordinate of the sensor and 0.5 represents the distance between adjacent points for all points n . (Note: n must be an integer and d must be a multiple of 0.5 because all points are 0.5 meters apart)
- d.) $\theta = \tan^{-1}((y_2 - y_1) / (x_2 - x_1))$ where \tan is the tangent function, (x_1, y_1) is the coordinates of the sensor and (x_2, y_2) is the coordinates of the target.

If determined that an object is obstructing the view of the target, the mobility control algorithm will then make an effort to relocate the range sensor to a position where the target is not obstructed. Using the results of the object modeling algorithm, the coordinates of the obstructing object are identified and possible new unobstructed positions are determined for the range sensor. The most suitable unobstructed position is then selected and used as the new location of the sensor. In most cases, the control algorithm will choose the unobstructed position closest to the range sensor's current location.

Hardware Implementation

I. Range Sensor

The range sensor we use in this project is the smart sensor 600 from SensComp. We choose this range sensor over others mainly due to its light weight (0.7 oz) and small dimensions ($0.95''$ by $3.4''$ by $3.4''$). It can detect object within 0.5 to 35 feet with an accuracy of 0.1% . The sensor has reasonable angular resolution with a 15 degree beamwidth at -6 dB. Its power consumption is $12\sim 48$ W during impulse transmission and $0.33\sim 1.32$ W without transmission.

The pin layout of the range sensor is shown as follows.

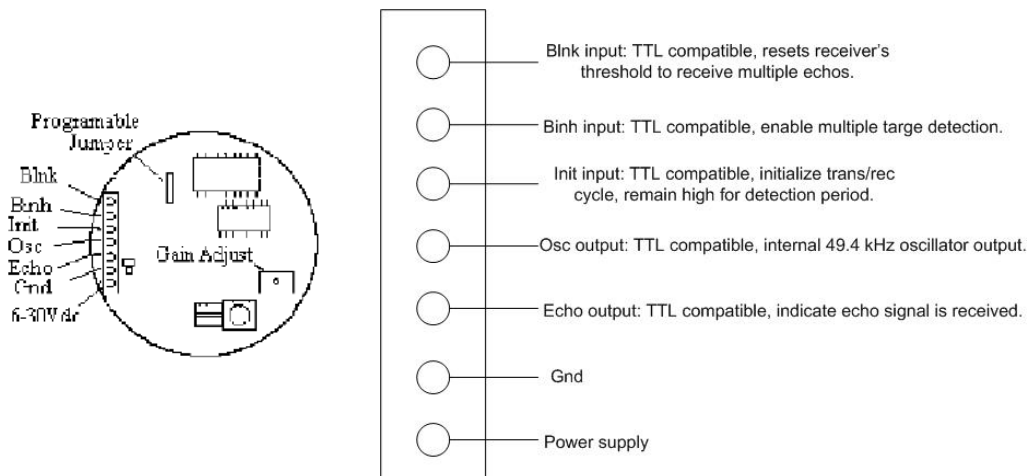


Figure 2. The pin layout of range sensor.

II. Single Echo Mode

In its single echo mode, the range sensor can detect a single object within its beamwidth. The basic signal waveform is shown in the next figure. The procedure is: 1) the sensor is power on for at least 5ms, and circuit starts up. 2) The INIT input assert high. The transducer produces 16 impulses at 49.4 kHz, and propagate through the air. 3) The receive input of the range control IC is inhibited by internal blanking for 2.83 ms after the INIT goes high in order to avoid the inherent ringing of the transducer being detected as a return echo. 4) Wait for the arrival of reflections. When it comes, assert ECHO output signal.

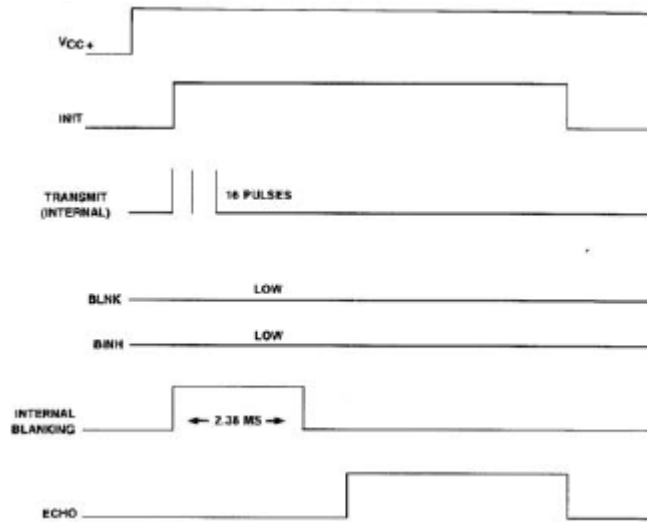


Figure 3. The signal waveforms when range sensor operates at single echo mode.

III. Multiple Echo Mode

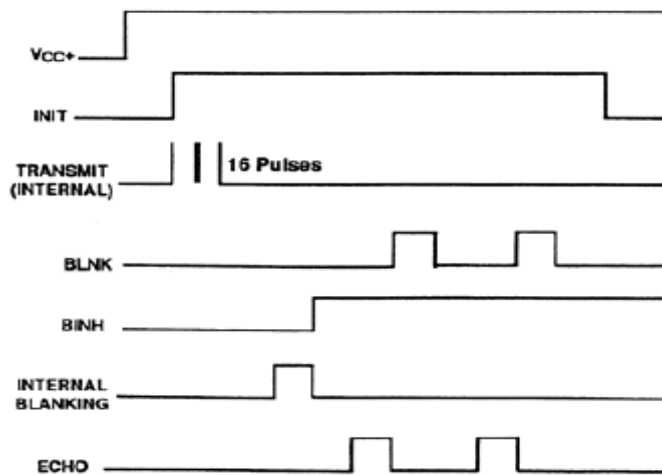


Figure 4. Signal waveforms when sensor operates at the multip echo mode.

In the multiple echo mode, the sensor is able to report the distance of multiple obstacles within the 15 degree beamwidth. The basic idea is to use the BLNK input to reset the ECHO output for the next return signal. The blanking signal is at least 0.44 ms to account for the 16 impulses and allow for internal delays. The procedure continues from the last step of the single echo mode operation. 5) Once the ECHO goes high, assert BLNK for about 0.44 ms. 6) BLNK goes low, and wait for the next echo signal to arrive. Go back to step 4. The signal waveforms are shown in Figure 4.

IV. Micra2

To connect the range sensor to a PC or stargate platform, we use the micro-controller (mica2) developed at UC Berkeley. It uses tinyOS and nesC. The following figure shows a pin layout of mica2.

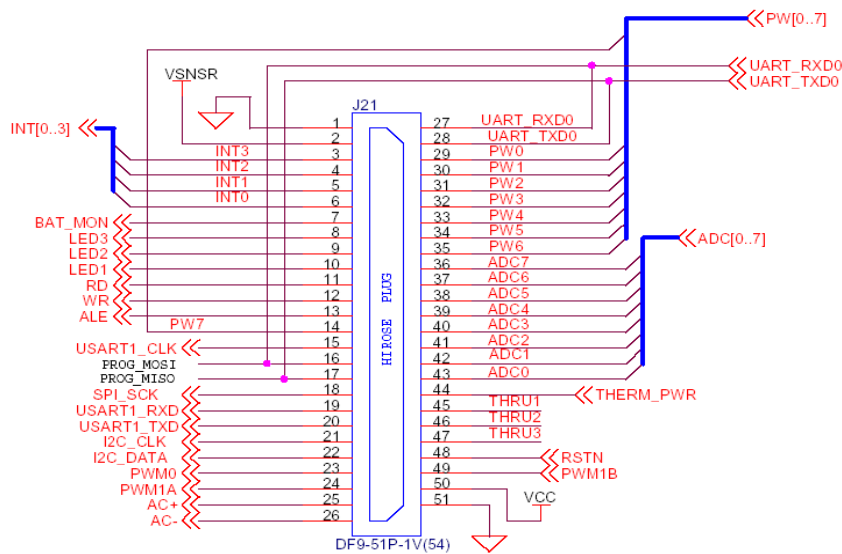


Figure 5. Mica2 pin layout.

For our project, we use mica2's LED1 to control the INIT input to the range sensor, and connect INT1 to ECHO. The implementation is described as follows. 1) Place the sensor at a certain position and toward a certain angel. 2) Turn the power on at the range sensor. 3) Assert LED1. At the same time, start a counter. As INIT is connected to LED1, INIT goes high as well. This starts the sensing. 4) Wait for the reflection. When ECHO asserts, an interrupt is received. Stop the counter. Record the value of the counter. 5) LED goes low. Move the sensor to a new location, or turn its angle to start another measurement. 6) Transmit recorded data to a PC or stargate.

Results and Discussion

I. Hardware Implementation

We connected the sensor to the mica2, and wrote some code in nesC trying to implement the algorithm introduced above. However, the effort has not been successful. First, there was error when downloading the compiled code to the mote. After we straighten this out, the mote doesn't work as we have projected. We didn't manage to get this part done mainly because we started the coding too late. We waited until the sensor's arrival at the eighth week. Instead, we spent time modeling the behavior of range sensor and produced mimic propagation matrix to test the mapping and control algorithm.

II. Mapping and Control Algorithm

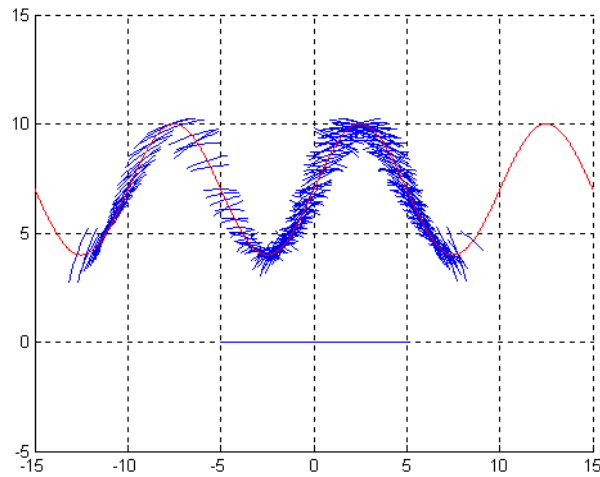


Figure 6. Mapping result: the red line is the real sinusoidal

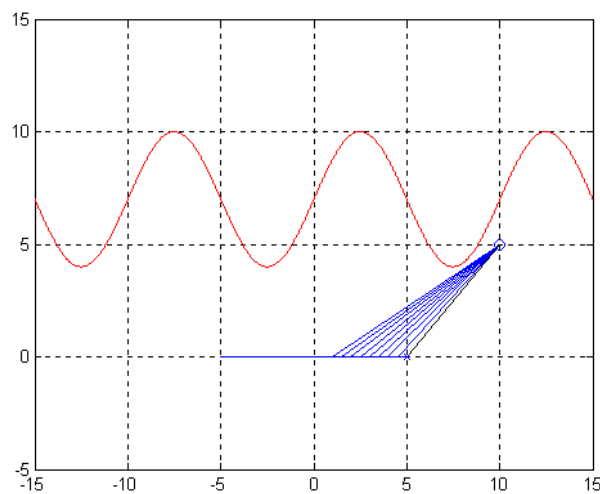


Figure 7. Control algorithm: the desired target is at (10, 5). The position where the sensor can see the target is connected by a line, and the closest one is indicated by a circle.

To model the behavior of the range sensor, we note two important characteristics. One is the angular resolution. With a 15 degree beamwidth (at -6 dB), the sensor will report only the closest object within its beamwidth. The other is the detection range. Beyond its detection range, the sensor is not able to tell whether there is an obstruction or not. We coded the algorithms both in matlab and C++. Figure 6 shows the mapping result when sensor can move between $[-5, 5]$ and the obstruction is a sinusoidal curve. Figure 7 shows the result of the control algorithm. It tells at which positions the sensor can see the target.

More examples are shown in Figure 8, 9, and 10.

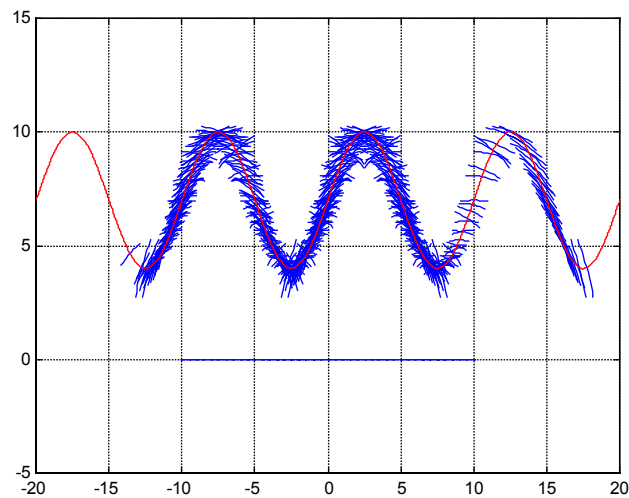


Figure 8. Sensor can move within $[-10, 10]$ on x axis.

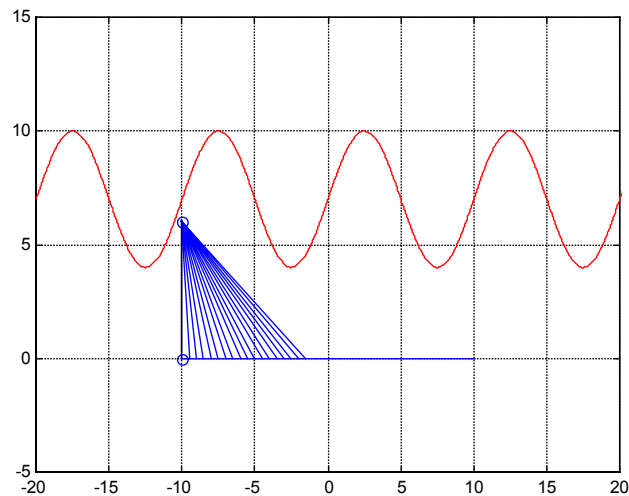


Figure 9. Monitor a target at $(-10, 6)$

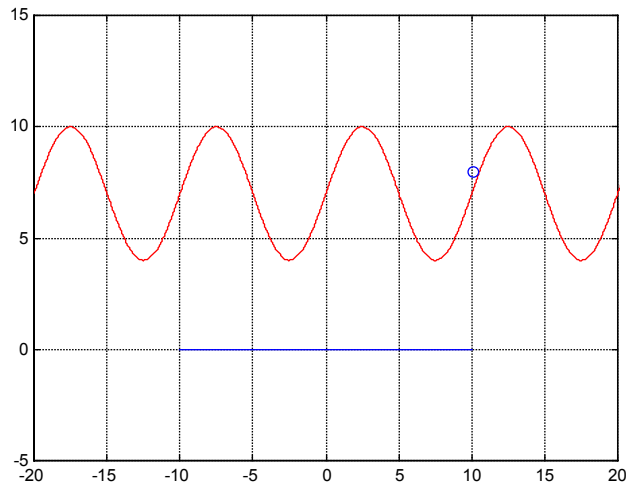


Figure 10. A target at (10, 7) is obstructed.

III. Discussions

Here, we bring up several issues that future workers may need to deal with when implement range sensors to enable self-aware locomotion. First of all, while sensors with narrow beamwidth have high angular resolution, they may fail to detect certain obstacle. Due to the narrow beam spread, the reflection surface needs to be close to a perpendicular position to the incoming beam so that the wave can be reflected and hit the sensor again. This is the problem with most laser range sensors. They have remarkable directionality, but the detection distance is in general a lot smaller than ultra sound sensors. Second, the multiple echo mode of the sensor can be used to increase the angular resolution. When operating at multiple echo mode, the sensor can report multiple objects present within the beamwidth. By comparing to measurements at adjacent angles, we can pin point how these objects are situated relative to one another. Third, the range sensor has both minimum and maximum detection distances. In some applications such as forest system monitoring, the sensor may be complete covered by fallen leaves. On these occasions, the sensors may falsely declare no obsacles. Appropriate strategies are needed to account for this issue. Third, when selecting the range sensor, power consideration has been a leading factor in favoring the small dimension and light weight. Also affected by power considerations is the sensing strategy to use. This includes questions such as what sampling rate shall we use to build the propagation matrix? (How many rows and columns shall we choose?) How often to update the matrix entries?

Conclusion

Despite certain limitations of range sensors (detection range, angular resolution etc.), they are able to provide more reliable and valuable obstacle information than static images taken by cameras. By carefully designing the algorithms, certain limitations can be alleviated. For example, using multiple echo mode, angular resolution can be

improved. It is demonstrated in this project that with distance ranging capability, it is possible for sensors to conduct self-aware locomotion and improve their sensing accuracy. Issues in sensor networks such as power consideration, sensor moving tracks place addition constraints on the algorithms.

References

- Robert K. Harle and Andy Hopper, "Building world models by ray-tracing within ceiling-mounted positioning systems," *UbiComp 2003*, LNCS 2864, pp. 1-17, 2003.
- Yang Li, Jason I. Hong and James A. Landay, "ContextMap: modeling scenes of the real world for context-aware computing," *UbiComp 2003*.
- SensComp 600 series smart sensor specifications.
- Jessica Feng et al., Obstacle identification and localization, CENS poster.
- Campbell Scientific SR50 ranging sensor manual,
<http://www.campbellsci.ca/CampbellScientific/Catalogue/SR50.pdf>.
- Linux platform, <http://www.linux.org/>.
- UC Berkeley Tinyos, <http://webs.cs.berkeley.edu/tos/>.
- ATmel AVR, <http://www.atmel.com/products/AVR/>.
- Discussions with Aman Kansal and Yen-Cheng Kuang.