Low Cost Ultrasonic Positioning System for Mobile Robots

Jan Dyre Bjerknes, Wenguo Liu, Alan FT Winfield and Chris Melhuish
Bristol Robotics Laboratories
Coldharbour Lane
Bristol UK BS16 1QY
jan.dyre.bjerknes@brl.ac.uk

Abstract

This paper describes a simple low-cost 2D and 3D positioning system for mobile robots. The system makes use of both ultrasound and radio frequency signals to achieve positional accuracies better than 1% of the tracked volume. Furthermore, we have integrated the positioning system into the simulation tools Player/Stage, in order to combine simulated robots with real robots in a ‘hybrid’ embodied simulation. The paper presents measurements from using the positioning system for both 3D tracking and 3D closed loop control.

1. Background

Absolute position measurement and tracking is often a requirement in experimental mobile robotics research, e.g. [Hereford et al., 2007], and may be problematical if multiple robots need to be tracked in real-time. A common approach is to make use of vision-based tracking systems, however such systems are typically complex and/or expensive, and demanding in 3D environments where one flying robot may visually occlude another.

In this paper we propose a solution to the problem of real-time position tracking of multiple robots in both 2D and 3D, by using a combination of ultrasonic signals for time-of-flight measurement, radio frequency clock synchronisation and trilateration. In common with vision based tracking systems, the system described in this paper requires a fixed infrastructure. However, unlike vision based systems there is no requirement for a central position-tracking server. Instead, each robot determines its own position autonomously. We argue that this is an advantage if the position tracking system is to be used as part of low-level closed-loop control, or as a basis for virtual sensors.

A similar approach to the one described here has been used for relative position tracking between robots [Spears et al., 2006]. Their approach placed three beacons on each robot, and necessarily requires more hardware per robot. Since our system is primarily directed toward 3D tracking of Helium aerobots with a very low payload mass (90g), then we clearly need to minimize the robot-mounted hardware.

This paper is organized as follows: Section 2 explains the mathematics behind trilateration that we have used in this system. Section 3 will describe the hardware in the system. This includes the base-station, the ultrasonic transmitters, ultrasonic receivers, and finally a small PIC micro-controller device placed on the robot for calculating the actual position. Section 4 will describe the 2D arena layout and more details on algorithms. Section 4 will also explain the integration with Player/Stage. Section 5 will explain the 3D arena layout, the 3D trilateration algorithms and show some results when the position tracking system is used for closed-loop control of a 90g flying robot. Section 6 concludes the paper and outlines further work.

2. Trilateration

Trilateration is the method of determining the position of an object when the distances to two or more fixed points are already known. There are several ways this can be calculated; in this project we have used Heron’s equation and standard trigonometry.

Consider the case in figure 1. We have two fixed points, 1 and 2, at known positions and a robot. If we know the distances from the robot to the fixed points, we can calculate the robot’s position, using Heron’s equation. We start by finding the semi-perimeter, Sp of the triangle described by the three points 1, 2 and the robot.

$$Sp = \frac{S + d1 + d2}{2} \tag{1}$$

Once we know the semi-perimeter we can calculate the area of the triangle using Heron’s equation:

$$Area = \sqrt{Sp \times (Sp - S) \times (Sp - d1) \times (Sp - d2)} \tag{2}$$

From the area we can then easily calculate the height of the triangle, h.

$$h = \frac{Area}{\frac{S}{2}} \tag{3}$$

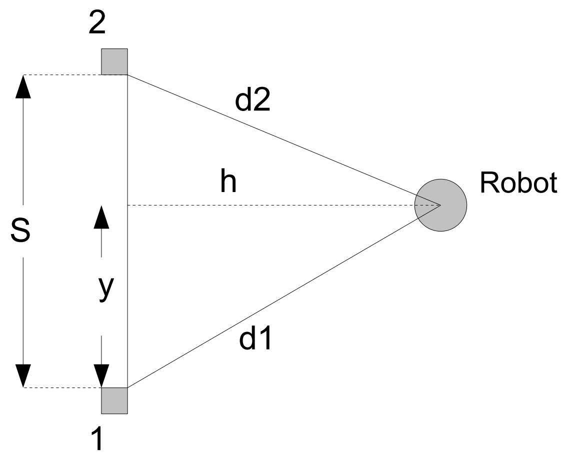

Figure 1: Based on the measured distances $d1$ and $d2$ we can calculate the value of h and y.

If the line between 1 and 2 is placed at position $x = 0$, h will be the x value for the robot's position.
The next step is to calculate y.

$$y = \sqrt{d1^2 - h^2} \tag{4}$$

We should make two observations so far: first, the value for y presupposes that 1 is located at $y = 0$, if not we would have to add the value for y in equation 4. We should also observe that this example has two solutions for the value of x, namely $x = h$ or $x = -h$. To fully determine the value of x we would need the distance to a third point. However, we have in the setup of the position system avoided this problem by always setting our fixed points at the outer edge of the area in which we want to locate a robot. We can therefore always locate a robot with two distance measurements for the 2D case, and three distance measurements for 3D.

2.1 *Measurement of distances and calibration*

To determine the values for $d1$ and $d2$ in the system, we use ultrasound and time-of-flight. We have placed ultrasonic transmitters at fixed positions in the environment and equipped the robots with an ultrasonic sensor. By synchronizing the ultrasonic pulse transmissions with a clock on the robot, the robot can accurately measure the time-of-flight to the transmitter. The clock is synchronized by a radio transmission.

However, it is important that the ultrasonic beacons are accurately placed in the environment. If a robot is in a position close to the line between two beacons, a small error in beacon placement will cause a large error in estimated position of the robot. Consider again figure 1, and assume that S is supposed to be 400 cm, but the beacons are misplaced so the distance between them is actually 405 cm. Further, let us assume that the robot is placed halfway between beacon 1 and 2, and the distance $h = 20cm$. This gives us $d1 = d2 = 203.48cm$. The semi-perimeter then becomes 403.48 cm and the area, $A = 7500cm^2$. From this it follows that $h = 37.49$ cm, an error of more than 17 cm. This error quickly becomes smaller as the robot is placed further away from the line S, and at a distance of 1m from the line S the error has been reduced to 4 cm. We can reduce this error even further by calculating the position using more than two beacons.

A similar argument can be made for temperature. If the positioning system is to be used in an environment with large changes in temperature this must be taken into account. The speed of sound changes by approximately 0.18% per $°C$. In an arena that is several metres long a temperature drift of a few degrees can give rise to measurement errors.

3. Hardware

The hardware in the position system is divided into two main parts. First, there is the hardware for sending clock synchronization signal and ultrasonic signals. This hardware is placed in the environment. Secondly there is the hardware on the robot, an ultrasonic sensor to detect the signals and a PCB with radio and a micro-controller to measure time delays and calculate the position based on the time delays.

3.1 *Hardware in the arena*

3.1.1 *Ultrasonic beacons.*

Both the 2D and the 3D system have 8 ultrasonic beacons. Even though we demonstrated above that two beacons are sufficient for calculating the x and y position in 2D, we have used 8 beacons to have good redundancy. If, for instance, people have to walk into the robot arena and therefore obscure one or more of the beacons, the robots will still be able to calculate their position. In addition, we saw above that a small error in beacon placement can give rise to huge position estimation errors. By using several beacons, the position can be estimated based on several different time-of-flight measurements and outliers can be filtered away. This increases the accuracy of the system. The 8 beacons are based around a 40 KHz transmitter from Murata. The beacons get their power and 40 KHz signal from the base-station.

3.1.2 *Base-station*

The base-station uses an 18F2520 PIC micro-controller from Microchip and is equipped with an Easy Radio LPRS radio. The radio is used for clock synchronization between the base-station and robots. The radio has an output that indicates if the radio is busy. This is a single bit output that is high during broadcast or reception. Figure 4 shows the time-line for a single positioning sequence. Initially, the Radio on the base-station broadcasts a single byte. During the broadcast, the busy-bit on the broadcasting radio and all receiving radios is set. When the broadcast is complete, the busy-bit clears and both the base-station and the robots start their timer. At the

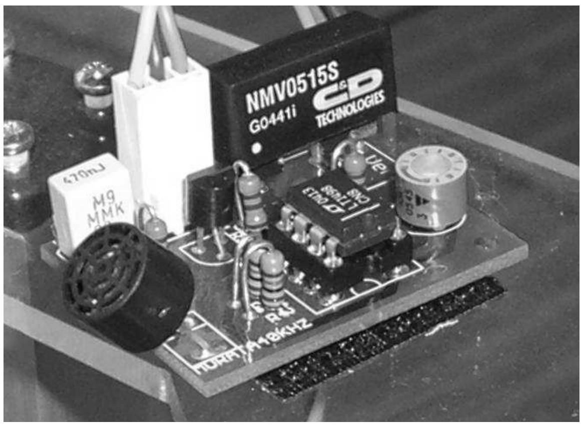

Figure 2: Ultrasonic transmitter unit.

same time, the base-station transmits the first ultrasonic pulse. Since the robots are now synchronized with the base-station, they can measure the exact time-of-flight. Furthermore, the other beacons will pulse at a predetermined interval and fixed order. By keeping track of time and received ultrasonic pulses the time-of-flight from the different beacons can be measured by the robots.

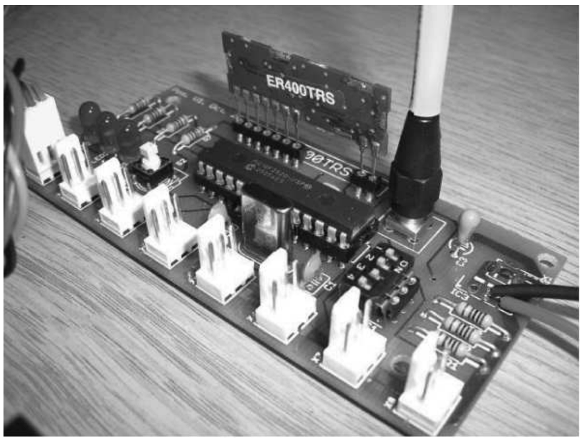

Figure 3: The base-station.

The time between each ultrasonic beacon transmission is different for the two arenas. In the 2D arena, the interval between the individual beacons is 50 ms, in the 3D arena it is 31 ms. The rationale for these values will be explained in detail in their respective sections.

3.2 Hardware on the robots

3.2.1 ultrasound sensor

On the robots there is a small circuit board with the ultrasonic receiver, a Murata 40 KHz sensor. The incoming signal is very weak, so it is amplified with a high gain amplifier. In the 2D arena the ultrasonic beacons are placed at the same height as the sensor on the robots. Due to the angular response of

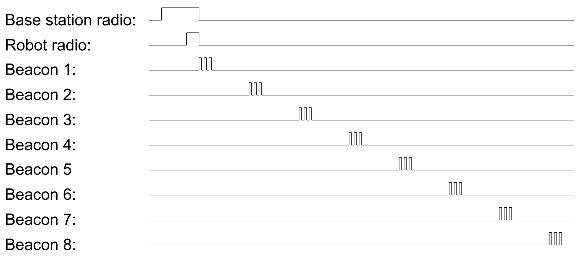

Figure 4: The time-line for a single positioning sequence.

the sensor [Murata, 2007], seen in figure 5, we have added a small reflective cone on the sensor. The cone ensures that the sensor can pick up all the beacons in the arena independent of the robot's position. A detailed image of the sensor with reflective cone can be seen in figure 7. For the flying robots no reflective cone is necessary. All beacons are placed on the floor and the sensors are facing downwards.

The Murata sensors are very sensitive at 40 KHz, and sensitivity falls off very rapidly at higher or lower frequencies. Acoustic noise in the environment does not appear to influence position measurements. However, on the 2D robots near-field interference from the wireless LAN-card does appear to affect the sensor. This problem has been solved by encapsulating the sensor electronics in a small MU-metal case.

On the 3D aerobots we have used two sensors placed 50 cm apart. This allows us not only to calculate the position of a robot, but also the robot's bearing in space.

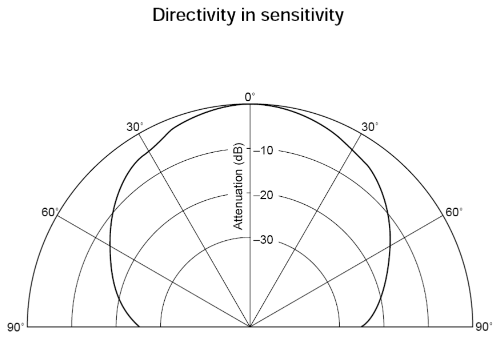

Figure 5: The ultrasonic sensor response as function of signal angle. The sensor is less sensitive when the signal comes from the side.

3.2.2 Robot PIC-boards

We have made two types of PIC-boards for the positioning system. For the 2D robots the PIC-board only needs to calculate the x and y position, as the robots already have ample computational power for motor control, reading other sensors etc. An 8 bit micro-controller was sufficient for this task.

On our 3D robots the micro-controller for the positioning system also handles all other sensors, drives all motors, does all the communication with other robots, and in general undertakes all computation for the whole robot. We need to be sure we have sufficient computational power on board and a 16 bit dsPIC 30F4011 micro-controller from Microchip was chosen in this case. The total mass including radio and two ultrasonic sensors is 16g. The increased computational burden of calculating heading in space (which actually means we have to do two complete x, y and z calculations) is another important reason for choosing a dsPIC on this PCB. An image of the aerobot controller can be seen in figure 11.

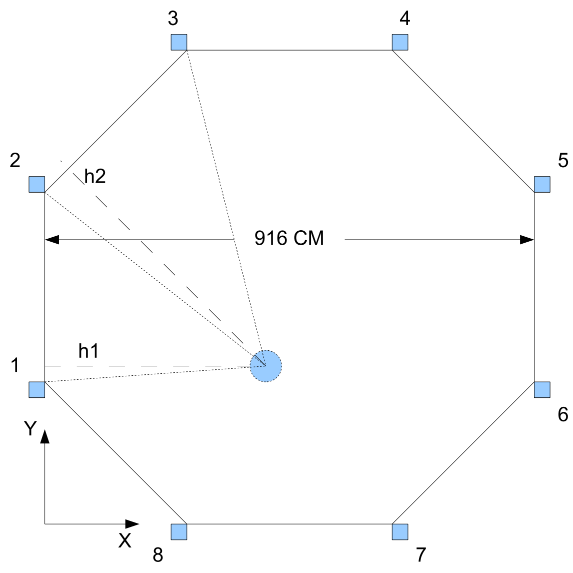

Figure 6: 2D arena layout.

4. 2D tracking

4.1 Arena layout and details of algorithms

The 2D arena has a layout as seen in figure 6. We see that the eight beacons are located at the eight vertices of the arena.

When the robot has measured the distance to beacon 1 and 2, it will calculate x and y as described earlier. When the distance to beacon 3 has been measured the robot calculates x and y again, this time based on the distances to 2 and 3. However, the result in this case is rotated 45° clockwise. To compensate for this, the x and y values must be rotated to the frame of the first measurements. This is done using the standard rotational matrix operation.

The time delay between each beacon is 50 ms. Even though it takes less than 30 ms for the sound to fly all across the arena, the value was chosen to make sure all reflections have died down. This means the robots can do a position update 20 times a second.

The x and y values calculated from two different pairs of beacons will usually not agree exactly. This is so partly because it is hard to place the beacons in exactly the correct position, and also because the ultrasonic signal is being attenuated over distance. Due to the signal attenuation and the fact that the sensor is a resonating device, it takes a little longer (typically $40 - 80$ uS) to detect the signal. This means that long distances tends to be measured as being slightly longer than they are. To avoid perceived drift in position when the robot is standing still, the x and y values are low-pass filtered, using a rolling average of the most recent 8 measurements.

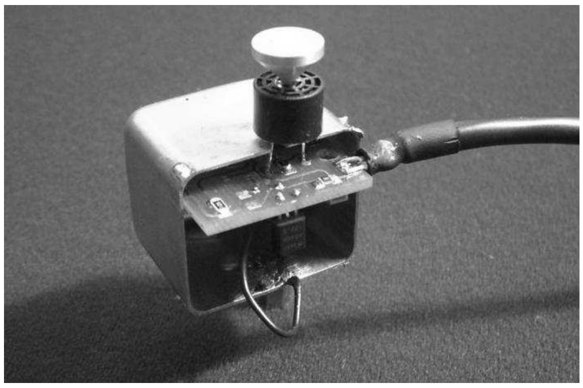

Figure 7: The sensor for the 2D LinuxBots. The sensor is encapsulated in MU metal to shield it from the Radio LAN card on the robot. Notice the grounding wire to the capsule. The small metal cone on top of the sensor works as a sound reflector and increases the sensitivity of the sensor.

In the 2D position system the x and y values are rounded to the closest centimetre. With the above algorithm there is no drift in position for a stationary robot. When the system is exactly calibrated with correct beacon placement and speed of sound, the absolute error in x and y is below 3 cm.

4.2 *Tracking the LinuxBots using position system via Player/Stage*

The most important application for the position system in our 2D environment is to track the wireless connected LinuxBots [Winfield and Holland, 2000] and log the positions of the robots so we can later replay and analyze experiments.
To this end, we have integrated the position system with the simulation tools Player/Stage [Gerkey et al., 2003] into our LinuxBots, where Player is a server that connects robots, sensors, and control programs over a network. Stage simulates a population of mobile robots, sensors and objects in a two-dimensional bitmapped environment. To track the positions for the real robot experiments, we only need to slightly change the control programs for each robot via Player. Figure 8 shows the flowchart to control and track one LinuxBot using Player/Stage and the position system at the same time. The client program gets the position data from the LinuxBots and also controls the robot to move around the arena via Player. Meanwhile, it sends the position data to Stage and draws the robot in Stage at the specific position. A screen shot from the tracking system is shown in Fig.9, where one robot is randomly moving around in our robot arena (size: 9.16×9.16

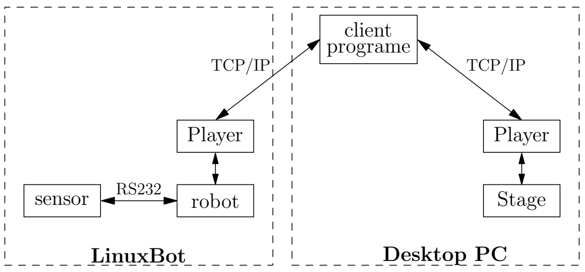

Figure 8: Using Player/Stage and position system to track the LinuxBots. The client program can be embedded in either the LinuxBot system or Desktop PC.

m) and at the same time avoiding the walls using three front mounted IR sensors. By using the "Show trails" function pro-

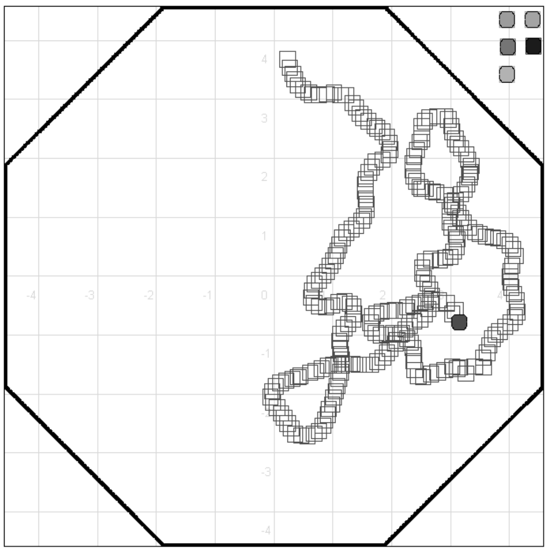

Figure 9: A screen shot from player stage as one robot is being tracked whilst performing a random walk.

vided by Stage, we can easily visualize the movement history for each robot in a user-friendly graph mode. The trails shown in Fig. 9 indicate that the position system can track the robots as they are moving around.

5. 3D tracking

In the 3D arena, the position system serves several purposes. First, the position system tracks and logs positions for the flying aerobots.

Second, since the position system is sufficiently accurate, the system can be used for closed loop control. If a robot has to move to a given position in 3D space, it does so using the positioning system to determine its current velocity and offset from the desired position. From these values a PD or PID controller controls the robot.

Finally, the 3D positioning system is being used as a virtual sensor. Since the robots have limited payload capacity several types of sensors cannot be carried onboard, but we can simulate the sensor onboard the robot, e.g. for obstacle avoidance or for wall following behaviour. This will be important for our aerobot experiments.

5.1 *Arena layout and details of algorithms*

Figure 10 shows the shape of the 3D arena and beacon placement. All the beacons are placed on the floor. In the 2D system the position is calculated continuously as the distances to the beacons are being measured. This gives a very fast update rate, and in the case of 2D tracking is easy to implement as the algorithm is identical for each measurement.

The 3D algorithm is more complicated due to the increase in number of dimensions. Therefore, the position of the robot is not being calculated each time a distance to a beacon is measured, but only when all the distances to beacons in the arena have been measured. This yields a slower update rate, but simplifies the algorithm and increases accuracy.

The time delay between beacons in the 3D system is 31 ms. It follows that a complete cycle with 8 beacons takes $8 * 31 = 248$ ms. This means that we can track a robot in 3D at 4 Hz.

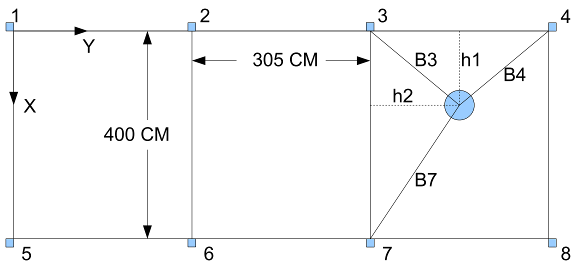

Figure 10: 3D arena layout. Z is not shown

Consider the case shown in figure 10. The Robot has measured the distance to all of the beacons, and calculates its position. This example will only involve the distance measurements from beacon number 3, 4 and 7, but the method is similar for all beacons. Based on the two measurements from beacon 3 and 4, the robot uses Heron's equation as described above and calculates the distance $h1$. If the robot was placed on the same height as the beacons, the distance $h1$ would be the x position for the robot. But as the robot has a position which is above the floor, the value $h1$ alone gives us no information about the position. Only when combined with either the value for $B3$ or $B4$ can the y value be calculated, using Pythagoras' equation. This means that in the 3D case, two beacons cannot determine the position in more than one dimension.

The same argument can be used for the distances $b3$ and $b7$. These two values determine the distance $h2$. By combining $h2$ with either $B3$ or $B7$ the x value is calculated.

From this the value for z can be calculated twice. Either by $z(1) = \sqrt{h1^2 - x^2}$ or $z(2) = \sqrt{h2^2 - y^2}$. Based on these equations and the placement of the beacons, there are clearly several ways to calculate the x, y and z values. To reduce noise and errors the x values are calculated four times in each cycle, by using beacon pair 1 and 5, 2 and 6, 3 and 7, and finally 4 and 8. The y values are calculated 6 times per cycle, by pairs 1 and 2, 2 and 3, 3 and 4 5 and 6, 6 and 7, 7 and 8. From this there are 10 different ways of calculating z.

Even though it is possible to calculate the y value from several other combinations of beacons (e.g by using 1 and 3, or even 1 and 4) we have not done so in this system. Using 6 estimates gives us the accuracy we need for our purposes.

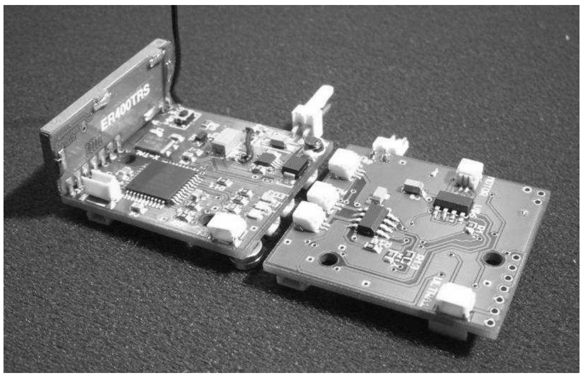

Figure 11: The PCB on the blimps. On the PCB to the left we can see the radio unit, the dsPIC, voltage regulator and some of the connectors for motors, sensors and battery. On the right side we see the reverse side of the PCB, showing the H-bridge drivers for the motors. Total mass including radio is 11.5g. The PCBs are attached to the robot carrying frame by means of a small permanent magnet.

Even though sensor noise is not a great problem, there is still a chance that one of the distances to a beacon is misread. This will influence the subsequent calculations of the position. By taking the median of all the values calculated for each axis, outliers are effectively filtered away.

As mentioned above the flying robots are equipped with 2 sensors that are placed 50 cm apart. By calculating the difference of x and y position for the two sensors, the robots' heading in space is easily found, using the standard C function atan2,

$$Heading = atan2(yDiff, xDiff); \quad (5)$$

returning a value between $+\pi$ and $-\pi$. We chose 50 cm distance between the sensors as a trade-off between a) having the two sensors as far apart as possible to achieve accurate heading measurement and, b) keeping them as close to the center of the robot as possible to avoid using long pieces of material to hold them in place. With a maximum payload of 90g every gram of material in the carrying structure makes a difference.

5.2 Computational power and code space

An important consideration is the computational power required for the tracking algorithm. All calculations of distances, positions and heading are done using 32 bit floating point arithmetic. At a clock frequency of 64 MHz, the complete calculation for x, y, z and heading takes less than 10 ms. The robot updates its position 4 times a second, thus the position system algorithm takes less than 4% of the computational power available in the micro-controller. This leaves ample computational power for the remaining tasks of the robot controller.

The micro-controller has code space for 16K words of instruction and out of this 3K words are used by the position code, leaving 13K. Out of 1500 bytes of RAM the position code uses approximately 100 bytes and requires a heap of 300 bytes.

5.3 Results, 3D tracking

In our 3D system repeatability of position estimation is the most important requirement. This is particularly important for closed loop control. I.e. if we want to drive the robot to different positions in the arena noise in the position will cause problems for the derivative part of a PD controller. We have measured the repeatability of x, y and z for several positions in the arena. A typical example is shown in figures 12, 13 and 14. Each of the figures shows the drift in measured position

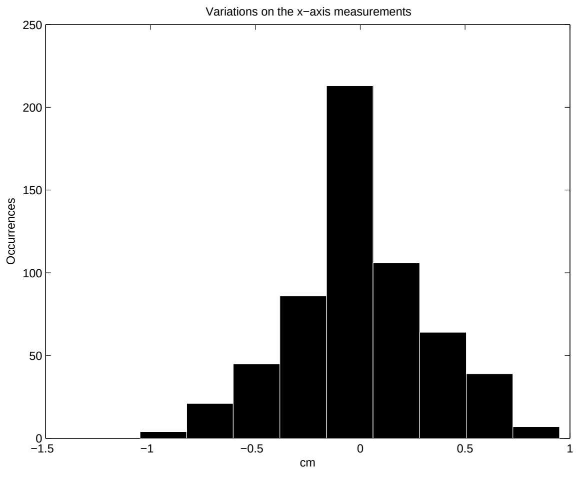

Figure 12: The variation in x for a fixed position.

for a period of 500s. In the case of the x-axis, we see that the variation is within $\pm$ 1 cm, and the standard deviation is 0.33 cm. Even though not yet tested in a control setting, we feel confident that this error is sufficiently small for our purposes.

We see similar reliability in the y axis, results shown in figure 13. The measured position drifts within $\pm$ 0.8 cm, with a standard deviation of 0.26 cm. We should notice that our arena is 900 cm in length and a drift in measured position of $\pm$ 0.8 cm is less than 0.1% of the arena size. Nonetheless, there is room for improving the result by using a more accurate synchronization unit.

The largest error by far is seen in the z-axis, where the variation for a fixed position is between $\pm$ 4 cm, with a standard deviation of 2.3 cm The reason for this large drift in error is

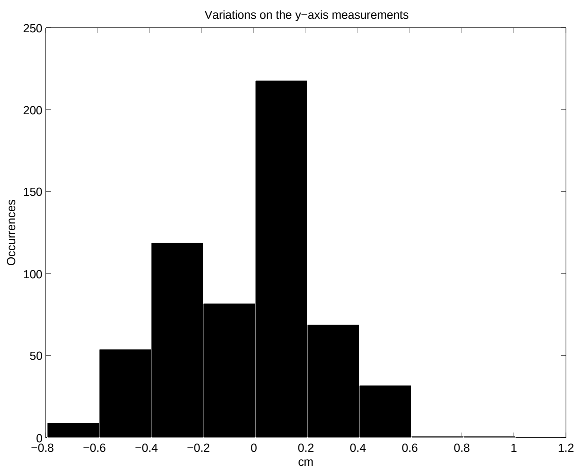

Figure 13: The variation in y for a fixed position.

due to the fact that the z value is calculated based on both x and y. The algorithm involves both multiplication and squaring of values, and this amplifies any small error in x and y axes.

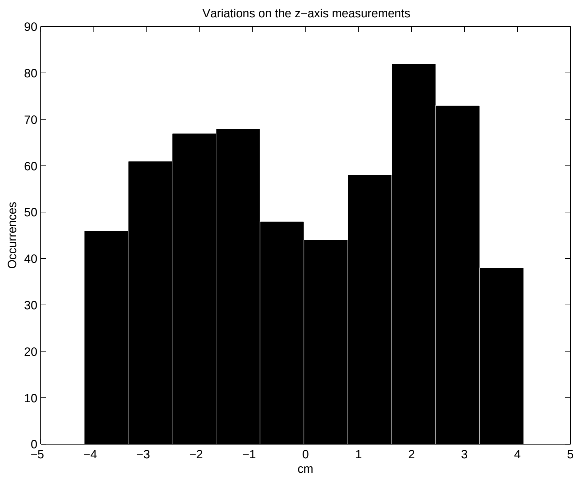

Figure 14: The variation in z for a fixed position.

When properly calibrated with exact beacon placement and exact speed of sound the overall absolute position error never exceeds 5 cm.

5.4 Results, 3D closed loop control

This large variation in z poses a challenge for our controller. The aerobot's maximum velocity in the z-axis is approximately 25 cm/s. Since the error drifts between $\pm$ 4 cm in a non-systematic fashion and the system updates the position at 4 Hz, there is a risk of the robot perceiving its velocity to be $8 * 4 = 32$ cm/s even though it is not moving at all. This is actually faster than the maximum speed of the aerobot. The PD-controller in the aerobots set $KP = 2$ and $KD = 5$. The consequence of this is that when the robot is relatively close to its target position, the noise in position will cause the controller to constantly drive the propellers in different directions with relatively high thrust. Even though the controller can maintain the robot in the desired position under such conditions, it will drain the batteries fast.

However, since we know that the robot can move at max 25 cm/s, and that it takes some time to accelerate to this velocity, we have used a 2^{nd} order IER filter with cut-off frequency of 0.2 Hz to filter the D term in the controller. The result is that the noise on the z-axis has much less influence on the controllers output.

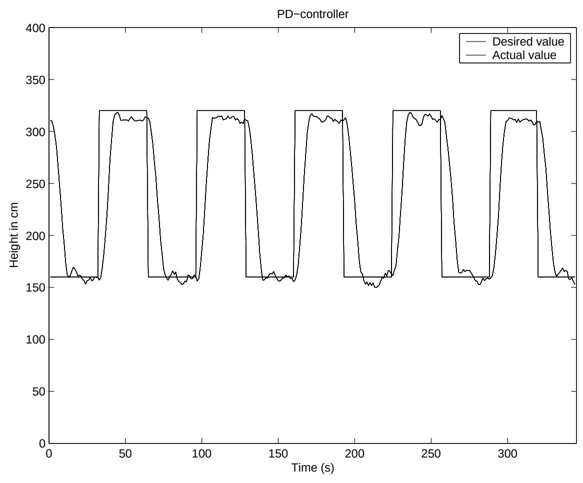

Figure 15: Steptest

Figure 15 shows the results from a step test of the controller. The aim of the controller is to maintain the aerobot at a height of 320 cm above the floor for 30s and then move the robot to a height of 160 cm, and then keep it there for another 30s, alternating up and down. Despite the noise on the z-axis, the robot follows the desired step height with reasonable accuracy.

We can see in figure 15 that the aerobot hovers around 160 cm when this is the desired height, but never really reaches 320 cm when that is desired. The explanation for this is that the robots are not perfectly neutrally buoyant but have a small negative bias. When working with Helium filled aerobots a slightly negatively bias is very helpful if the robot runs out of battery power or for some other reason stops driving its motors. Nonetheless, the controller is sufficiently accurate for our purposes.

We also tested the height control of the aerobots using a sine wave as input. Figure 16 shows the result when a robot tries to move up and down with a periodic time of 40s and peak-to-peak height of 160 cm. We see that the robot follows its desired trajectory with good accuracy.

6. Conclusion

This paper has outlined the design of a 2D and 3D position tracking system for multiple mobile robots. The hardware

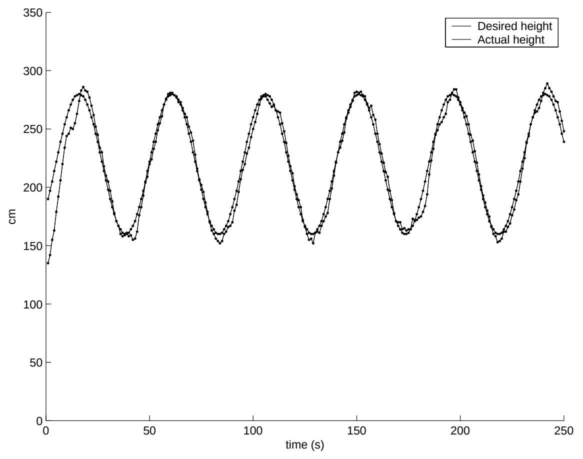

Figure 16: Sinewave

of the system requires both fixed beacons and robot-mounted position tracking 'sensors'. This paper has described low-cost electronics hardware for these system components. In particular the demanding requirements of position tracking of a Helium aerobot have been met by a position tracking sub-system (ultrasonic sensor, radio device and micro-controller circuit) with a total mass of only 16g.

One important benefit of this system compared to e.g. visual tracking systems, is that in our system the robots calculate the position themselves. This means that the system can easily track a large number of robots, as an increasing number of robots does not increase the computational load on the system. In addition, since the robots themselves compute their position they can use this information for control, without having to have the position information sent to them over radio.

Integration of the position tracking system with Player/Stage has provided us with the novel possibility of running 2D multi-robot simulations in which some of the robots in the swarm are real physical robots, while the rest exist only in simulation. Yet the real and simulated robots interact with each other seamlessly. We believe this to be a valuable tool both because it allows larger swarms to be simulated, and because the real robots ground the simulation in real-world physics.

Repeated position tracking measurements have been demonstrated to show a variation of less than $\pm$ 1 cm for x and y axes, and $\pm$ 4 cm for the z-axis, in 3D. Given that the 3D arena measures 4m x 4m x 9m; this represents a worst case error (on the z-axis) of about 1%. Notwithstanding the worst case, we have demonstrated effective closed-loop control in the z-axis. In conclusion, this level of performance is very creditable given the simple low-cost hardware employed, which fully meets the experimental need.

6.1 Further work

The most important improvement for this system is to add a temperature sensor. As the temperature in the Bristol Robotics Laboratory changes by more than 10°C throughout the year, the systems needs re-calibration with different air temperature. This is clearly impractical. The best solution for this is to add a temperature sensor to the base-station. The 18F PIC micro-controller has the necessary input for such a sensor, and is already broadcasting a byte to the robots for synchronization purposes. This synchronization signal could then contain the temperature information and the robots change their constants accordingly.

Acknowledgements

The authors would like to thank colleagues Ian Horsfield and Alexander Lenz for many useful and helpful contributions in the design of these two position systems.

References

Gerkey, B., Vaughan, R., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. *Proceedings of the International Conference on Advanced Robotics*, pages 317–323.

Hereford, J. M., Siebold, M., and Nichols, S. (2007). Using the particle swarm optimization algorithm for robotic search applications. *Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)*, pages 53–59.

LPRS (2003). Lprs datasheet. easy-radio ER400TRS transceiver. www.lprs.co.uk.

Microchip. (2006). dsPIC30F4011/4012 datasheet. www.microchip.com.

Murata (2007). Murata MA40S4 datasheet. search.murata.co.jp.

Spears, W. M., Hamann, J. C., Maxim, P. M., Kunkel, T., Heil, R., Zarzhitsky, D., Spears, D. F., and Karlsson, C. (2006). Where are you? In Sahin, E., Spears, W. M., and Winfield, A. F. T., (Eds.), *Second International Workshop on Swarm Robotics at SAB 2006*, volume 4433, pages 129–143, Berlin, Germany. Springer Verlag.

Winfield, A. and Holland, O. (2000). The application of wireless local area network technology to the control of mobile robots. *Journal of Microprocessors and Microsystems, 23/10*, pages 597–607.