THE INSTITUTE FOR SYSTEMS RESEARCH

ISR TECHNICAL REPORT 2007-13

Pursuit Techniques on Pioneer Robots

Thomas Gehrels

The Institute for Systems Research

A. JAMES CLARK SCHOOL OF ENGINEERING

ISR develops, applies and teaches advanced methodologies of design and analysis to solve complex, hierarchical, heterogeneous and dynamic problems of engineering technology and systems for industry and government.

ISR is a permanent institute of the University of Maryland, within the A. James Clark School of Engineering. It is a graduated National Science Foundation Engineering Research Center.

www.isr.umd.edu

Pursuit Techniques on Pioneer Robots

By Thomas Gehrels

In collaboration with Scott Watson and Jansen Sheng

Institute for Systems Research

Research Experience for Undergraduates Program

Under the Direction of Dr. Krishnaprasad

August 3, 2007

Abstract

This summer's research goal was to investigate different types of pursuit games. We seeked to develop feedback controls laws that will realize strategies such as classical pursuit, constant bearing pursuit, and motion camouflage. The latter being a strategy used in nature, for example, when hoverflies chase possible mates or when dragonflies engage in aerial territorial battles. (Echolocating bats hunt insect prey using a constant absolute target direction (CATD) strategy which is geometrically indistinguishable from motion camouflage.) These algorithms were implemented on the Pioneer robots using the high level motion control language MDLe. We also implemented an extended Kalman filter to improve the position estimation provided by the Cricket sensors.

Table of Contents

1. Introduction

After thousands of years of evolution, nature has been able to develop optimal and efficient strategies for pursuit and evasion. One such strategy is motion camouflage, which is used by hoverflies to chase possible mates and dragonflies during aerial territorial battles. Echolocating bats also hunt their prey using a geometrically equivalent version of this pursuit. Following this strategy minimizes the lateral movement of the pursuer in the perspective of the evader. Likewise, this strategy can be used by the evader in an attempt to escape from a predator without being seen. This study hopes to recreate these strategies with feedback control laws in a laboratory environment using robots. Methods of accurate position estimation including Cricket, Bancroft's Algorithm, and Kalman filtering, will also be explored to supplement the control laws.

1.1 Pursuit Strategies

Classical pursuit is a pursuit technique in which the pursuer's instantaneous velocity vector is always pointed directly towards the evader at every instant in time. It has been shown that when the evader moves in a straight line, the movement of the pursuer traces a logarithmic path[6]. One interesting problem that uses classical pursuit is the "Three Bugs Problem." The "bugs" represent moving particles, each of which is acting as both a pursuer and an evader. Despite its name, the simplest case is when there are four "bugs," or particles. In this scenario each particle begins at one vertex of a polygon with as many sides as there are particles, so in this case a four sided square. The particles, which we can denote Bug 1 through Bug 4, will pursue each other such that Bug 1 pursues Bug 2, who is pursuing Bug 3, who is pursuing Bug 4, who finally is pursuing Bug 1. The aim of this problem is generally to determine either the distance traveled or time elapsed before mutual capture occurs, or to trace the path of the particles. The distance traveled by each particle for a generic problem involving n bugs on an n-sided polygon with unit length sides is equal to

$$d = \frac{1}{1 - \cos\left(\frac{2\pi}{n}\right)}. \qquad (1)$$

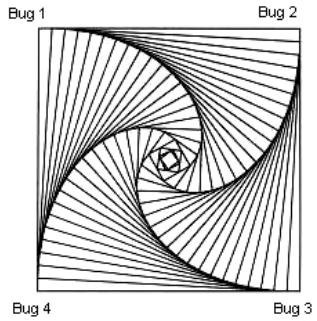

Figure I. The path of the four bugs problem.[9]

This information, along with the velocity of the bugs can be used to determine the time elapsed before mutual capture[9]. Figure I shows the path that will be traced by each of the four particles (the dark lines), along with the direction of their velocity at interval time instances (the lighter lines).

Constant bearing pursuit is a technique that is time optimal in the case that the motion of the evader is limited to a straight line. This technique involves nothing more than determining the closest point at which the pursuer can intersect with the path of the evader, and traveling in a straight line towards this point. This technique is good when the evader has no control over its movement aside from setting their initial velocity. For all other instances, such as when the evader is able to change its direction or velocity, this technique is not time optimal.

Motion camouflage is a pursuit technique that has been seen being practiced by a variety of insects and predators in nature[1]. When using the technique the pursuer and evader remain at the same bearing relative to a fixed point. The fixed point chosen is usually a notable object in the distance, or in the case of some pursuers, a theoretical point at an infinite distance. From the evader's point of view the pursuer will not have any lateral movement relative to the fixed point, and its only motion will be along the vector that joins the two. This proves to be an effective pursuit strategy when the evader is a species with compound eyes. In this case, the evader is very sensitive to optical flow, or lateral movement, but not sensitive to looming, the component of the relative movement between pursuer and evader along their joining vector. This technique has been observed between hoverflies, dragonflies, and a geometrically indistinguishable technique was also observed in use by echolocating bats when hunting insects[1].

The implementation of motion camouflage on the robots used in the lab requires a set of control laws that result in the correct motion based on the position and velocity of the pursuing robot and the evading robot. Since the robots in the lab are not equipped with cameras or other

hardware that could be used for locating the other robot, the robots are set up to pass their position and velocity to the other robot instead. The velocity of the pursuer is defined as unit speed, and the evader has speed $v < 1$. Justh & Krishnaprasad (2006) define the system of equations in terms of three vectors, the **r** vector, which is the position vector of the robot, the **x** unit vector, which points in the direction tangent to motion, and the **y** unit vector, which points normal to the direction of movement.

For the pursuer, these vectors will have subscript "p" for pursuer, and for the evader they will have subscript "e." Based on these three vectors, the motion of the pursuer is defined by the equations

$$\dot{r}_p = x_p , \tag{2}$$

$$\dot{x}_p = y_p u_p , \tag{3}$$

$$\dot{y}_p = -x_p u_p , \tag{4}$$

where u_p is the steering control for the pursuer. Similarly, the motion of the evader is defined by the equations

$$\dot{r}_e = v x_e , \tag{5}$$

$$\dot{x}_e = v y_e u_e , \tag{6}$$

$$\dot{y}_e = -v x_e u_e \tag{7}$$

where u_e is the steering control for the evader, and $0 \le v < 1$. Finally, the baseline vector, or the vector pointing from the evader to the pursuer is defined as

$$r = r_p - r_e . \tag{8}$$

Based on these equations, Justh & Krishnaprasad show that the steering control for the pursuer is given by

$$u_p = -\mu \left(\frac{r}{|r|} \cdot \dot{r}^{\perp} \right), \tag{9}$$

where the notation $\dot{r}^{\perp}$ denotes the $\dot{r}$ vector rotated counter-clockwise by $\pi/2$ radians, and the gain constant has value $\mu > 0$. The only control laws on the robots are the velocity v and the angular velocity ω. The velocity of the pursuer is defined as unit velocity, so this control is a constant. The equation for the angular velocity is derived from the velocity equation

$$v = \omega \times r, \tag{10}$$

which when differentiated gives you the equations

$$a = \omega \times v + \alpha \times r\,. \tag{11}$$

Since velocity is constant, the only acceleration possible is in the radial direction, so the value of $\dot{\omega}$ must be equal to zero because its cross product with *r* would point in a nonradial direction in any other case. The acceleration value a will thus be given by

$$a = \omega \times v\,. \tag{12}$$

We can now calculate the acceleration based on the equations 2, 3, and 4. The acceleration will be given by the derivative of velocity with respect to time, as

$$\frac{d}{dt}(\dot{r} = x)\text{, which gives} \tag{13}$$

$$a = \ddot{r} = \dot{x}\,. \tag{14}$$

When we substitute in equation 3 and 12, we get

$$\omega \times v = y_p u_p\,. \tag{15}$$

Since the velocity always has unit magnitude and points in the same direction as x_p, and y_p also has unit magnitude but points perpendicular to velocity, the angular velocity must point normally to the xy plane and have magnitude

$$\|\omega\| = u_p. \tag{16}$$

1.2 The Extended Kalman Filter

The Kalman filter was developed by Rudolf Kalman as a recursive filter that can estimate the current state of a system based on its previous state and intermittent noisy measurements. This filter works for linear systems by linearly combining the current state and the control in order to determine the next state[10]. For our purpose, the Kalman filter was used to calculate the position and heading (direction) of each robot based on noisy cricket measurements and odometry control data. Our state variable matrix **x** is defined as

$$\mathbf{x}_k = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \tag{17}$$

where k denotes time and our control variable matrix **u** is defined as

$$\mathbf{u}_k = \begin{bmatrix} d\mathrm{r} \\ d\theta \end{bmatrix}. \tag{18}$$

Based on these variables the state at time k+1 will be equal to

$$\mathbf{x}_{k+1} = \begin{bmatrix} x + d\mathrm{r}\cos(\theta) \\ y + d\mathrm{r}\sin(\theta) \\ \theta + d\theta \end{bmatrix}. \qquad (19)$$

Unfortunately this is not a linear combination, and thus the linear Kalman filter will not work for this system. Instead a filter that is capable of handling nonlinear systems will be needed and the extended Kalman filter has just this quality[8]. This filter works much in the same way the Kalman filter does, but the state estimate for time k+1 needs not be a linear combination of the current state and controls, rather it can be any nonlinear combination[10]. Our earlier equation for the estimated state can be rewritten in terms of the state and control matrices as

$$\mathbf{x}_k = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k-1} + \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d\mathrm{r} \\ d\theta \end{bmatrix}_k. \qquad (20)$$

Note that we are now assuming we are at time k, and that the last state was at time k-1. A matrix $\mathbf{F}_k$ is calculated by taking the Jacobian of the above function in terms of the matrix $\mathbf{x}_k$. This matrix will thus have the value

$$\mathbf{F}_k = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & 1 & \cos(\theta) \\ 0 & 0 & 1 \end{bmatrix}. \qquad (21)$$

This matrix is used to calculate the error covariance matrix $\mathbf{P}_k$ by the formula

$$\mathbf{P}_k = \mathbf{F}_k \mathbf{P}_{k\text{-}1} \mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k, \qquad (22)$$

where $\mathbf{Q}_k$ is a 3x3 matrix whose entries are the covariance of the noise coming from our control measurements. Next we receive the (noisy) position data z_k, and calculate the innovation $\mathbf{y}_k$, or the difference between the position data and our estimated position, by

$$\mathbf{y}_k = \mathbf{z}_k - \mathbf{x}_k. \qquad (23)$$

A matrix $\mathbf{H}_k$ is now calculated, with its value being the Jacobian of the estimated position data based on the state. In our problem, the estimated position data is equal to the state, so $\mathbf{H}_k$ is just a 3x3 identity matrix. The Kalman gain $\mathbf{K}_k$ is then calculated using $\mathbf{H}_k$ by

$$\mathbf{K}_k = \mathbf{P}_k \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^T + \mathbf{R}_k)^{-1}, \qquad (24)$$

where R_k is a 3x3 matrix whose diagonal entries are the covariances of the noise from the measurement data. This matrix is used to update the state of the system by the equations

$$\mathbf{x}_k = \mathbf{x}_k + \mathbf{K}_k \mathbf{y}_k \,. \tag{25}$$

Finally, the error covariance $\mathbf{P}_k$ is updated based on the new value of $\mathbf{K}_k$ such that

$$\mathbf{P}_k = (I - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k \,, \tag{26}$$

where I is a 3x3 identity matrix. Since the filter runs recursively, the values of these matrices is next used to calculate the state at the next time instant.

2. Methods and Materials

2.1 ActivMedia Pioneer Robots

Figure II. A photo of both Pioneer robots used in the laboratory.

The Pioneer 2 and 3 robots, named Lola and Genghis respectively, used in the laboratory were built by ActivMedia Robots. The robots are approximately 50cm x 49cm x 26cm, can carry payloads of up to 30kg, and can reach speeds of up to 70 centimeters per second. Three fully charged batteries will power the robot for three to six hours and can be recharged in about two and a half hours. Each robot is equipped with an onboard computer that runs Debian Linux. Users are able to log on to the system directly by attaching a monitor and keyboard to the robot, or remotely by connecting to the wireless bridges installed on the Ethernet ports. Multiple sensors are installed on each robot including a set of eight sonars located on the front of the robot and an odometry device that determines how far each wheel has moved. In addition, Lola has both front and rear bumpers attached to prevent possible collisions between objects or other robots. Scripts are programmed in the extended motion description language (MDLe) based on C++. MDLe was developed at the University of Maryland for the purpose of providing a universal platform to control robotic systems. Communication between robots, which is needed

to pass the necessary position and velocity data into the control laws, is enabled through a CORBA interface.

2.2 Cricket Devices

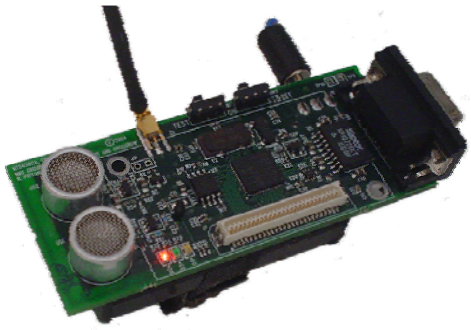

Figure III. A picture of a Cricket device used by the robots.

This research depends on a versatile acoustic location system called Cricket because there is not another reliable system that can be implemented inside of the lab. Systems such as GPS cannot be used because it does not operate correctly indoors. The system is built from collections of "Cricket units" that are arranged in some network of beacons and listeners. Each Cricket unit has the ability to send an acoustic pulse at 40 kHz and detect sound at the same frequency. Additionally the units can communicate with each other with a small digital radio in conjunction with a simple Atmel microcontroller[7]. The primary function of the units is to measure how long sound takes to travel from the client unit to beacon units. Using one of many algorithms in conjunction with multiple simultaneous range measurements, the location of a client can be determined. Bancroft's algorithm and Kalman filtered range measurements are the two primary techniques used for positioning in the Intelligent Servosystems Laboratory (ISL).

The current setup at ISL has an array of cricket beacons on the ceiling and client crickets placed on the robots. The setup is able to determine the location of any client within the coordinate system of the lab. Each robot is given two clients so that heading information can be easily determined by calculating the line between the two units; because of the magnetic interference from the iron reinforcement in the floor of the lab, electronic compasses proved inadequate for heading measurements. One predetermined leading client gives a radio frequency (RF) "sync pulse" and all other clients are programmed to use this pulse to initialize time. At predetermined time intervals after the pulse, each client performs its task. Listener beacons on the ceiling immediately start a timer and begin listening for the 40 kHz ultrasonic pulse[7]. All the

clients have a time slice to send out their acoustic pulse to the beacons. When the sound is received by a beacon the timer value is sent back to the client by RF. This process runs to completion before the next beacon is scheduled to run the same process. In this way each beacon collects three "time of flight" measurements that are used to calculate range and then position via various algorithms.

There are several issues that affect the accuracy of any cricket implementation. They include temperature gradients in air, the structure and sensitivity of the microphone, the structure and power of the speaker, significant movement of the cricket units in space, and multipathing of the sound waves. Temperature gradients cause the path that the sound takes from point A to point B to refract or trace a curve depending on the severity of the temperature differences. This lengthening of path is hard to account for without a thorough knowledge of one's environment. In the changing conditions inherent in buildings with HVAC systems, this information can change dramatically from hour to hour. Since temperature does not contribute to more than a few millimeters of range measurement error for the distances used in the lab, the data is still useable.

Because the microphone is encased in a metal can with a metal mesh protection, this part has a poor pole plot[7]. Ideally a microphone for this kind of system would be able to detect a sound coming from all directions with equal sensitivity. As you can see from the pole plot the sensitivity is strongest directly in front of the microphone and becomes unusable outside of a 45 degree cone.

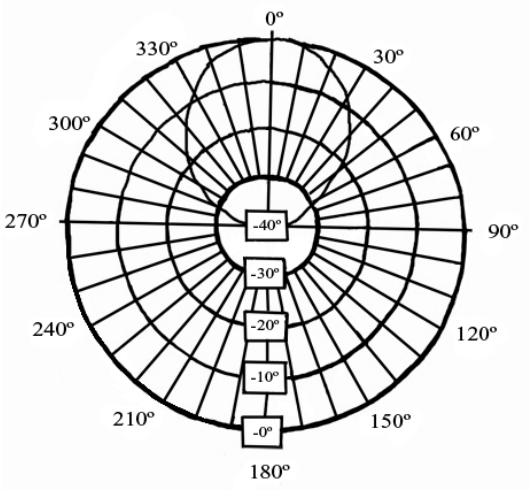

Figure IV. Pole plot for Cricket device

The element that actuates the sound is also prone to problems similar to those of the microphone. It is encased in the same metal can with mesh protection as the microphone. Sound most easily passes through the mesh parallel to the sound source and is reflected off the sides of the metal can. This setup can be improved by using a speaker with higher dB output. Ideally the intensity of sound produced by this element would have equal intensity at any angle from a fixed distance. A speaker with a hemispherical shape would significantly improve performance outside of the 45 degree cone.

Range measurements made by the cricket system are ideally suited for stationary objects. A range measurement that may have been accurate for a stationary robot may become inaccurate and thus unusable for a moving robot in a matter of milliseconds. This is because the ultrasonic pulse takes time to travel through the air, so by the time the range has been calculated the robot will have already moved. The time range that a measurement remains acceptable depends heavily on the speed of the robots and the frequency with which range measurements are made[2]. These movement problems become more complicated when multiple ranges are collected. Because sound takes time to travel through air, three different range measurements could be measuring three slightly different client positions depending on the configuration. The nature of the cricket system also requires that the measurement frequency be decreased when more range measurements are required. This is because the radio communications between the beacons and the clients must take turns to complete.

Multipath is another significant difficulty introduced by acoustic location systems. This problem occurs when a listener cricket unit detects a 40 kHz pulse that has bounced off any surface instead of traveling directly from the source[7]. A common symptom of this problem is that an unrealistically short range measurement is taken. After a cricket unit starts listening for the signal there is no way of differentiating the real signal from a bounced signal with the current design. Mulitpath issues can manifest themselves as intermittent incorrect range readings. Determining multipath as a problem can be difficult to do. One way to alleviate these problems includes alternating between multiple frequencies. If a measurement is made using a 40 kHz signal then the microcontroller can do an FFT on the incoming signal and send a range measurement only when a 45 kHz pulse is heard. Depending on the amount of reverberations and time for the signal to adequately decay, any number of frequencies can be used within the limitation of the signal. Additionally, sound can be pulsed with different duty cycles. The

microcontrollers onboard the cricket units could be programmed to recognize the difference between a 1ms pulse and a 2ms pulse and respond according to the same alternating scheme as with the previous frequency example.

There is enough digital processing capability in the Atmel microcontroller such that some basic pre processing can be done on the range data before data is sent to the host computer through the RS-232 serial port link. One of the more sophisticated preprocessing steps that can be performed is Bancroft's algorithm. Bancroft's algorithm takes three range measurements from known locations and uses least squares estimation to determine the approximate location of the beacon cricket unit to about 3cm of accuracy[8].

3. Results and Discussion

3.1 Cricket Data

The project was carried out in several stages. In the first step, the control laws for the pursuit strategies were coded onto the robots and executed. At this stage, the Cricket system with Bancroft's algorithm was the only reliable means of determining the position of a robot in the laboratory. However, the resulting data was not frequent enough to provide reliable inputs into the control laws for the robots. Data would only be collected every second at most and often times there would be no good data for up to four or five seconds. Another issue was that the Crickets generated a significant number of outliers. The result of these two factors was that the position estimates for the robots were not continuous and were likely to contain large errors. Analysis of the Cricket data shown in Figure V for a stationary robot accumulated over several thousand data points returned a standard deviation of 10 centimeters in the x direction, 4 centimeters in the y direction, and 0.4 radians for the orientation. The intermittence and variation in data led to constant over-steering or steering in the wrong direction when attempting to run any pursuit algorithms.

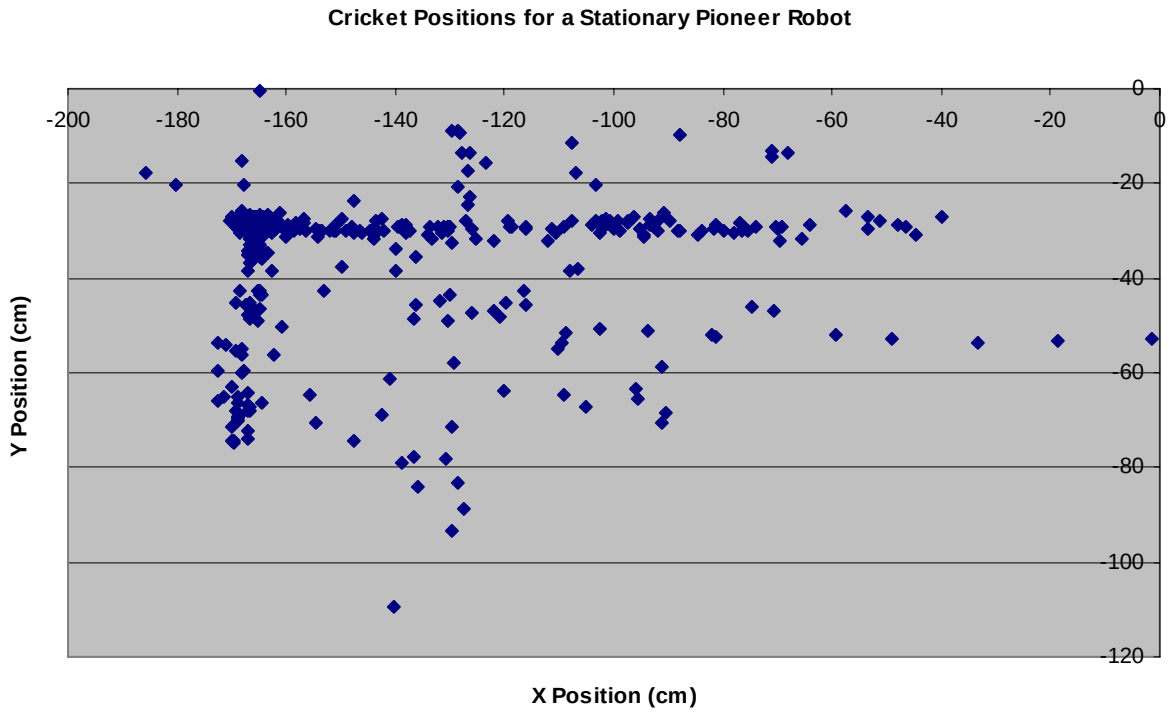

Figure V. A plot of the Cricket positions for a stationary robot

3.2 Extended Kalman Filter

The disappointing results of position estimation with only Cricket led to the next step, to implement a Kalman filter in order to improve the estimates of the robot's position. The first version of the filter combined Bancroft's algorithm, which was already hard coded onto the Crickets, with the onboard odometry device. The results of the filter were much more practical as the data points were more frequent and converged to a proper value. The filter was resistant to the outliers that Cricket measured and provided results several times faster. The fluctuation of the output was about one centimeter in the x direction and about a third of a centimeter in the y direction.

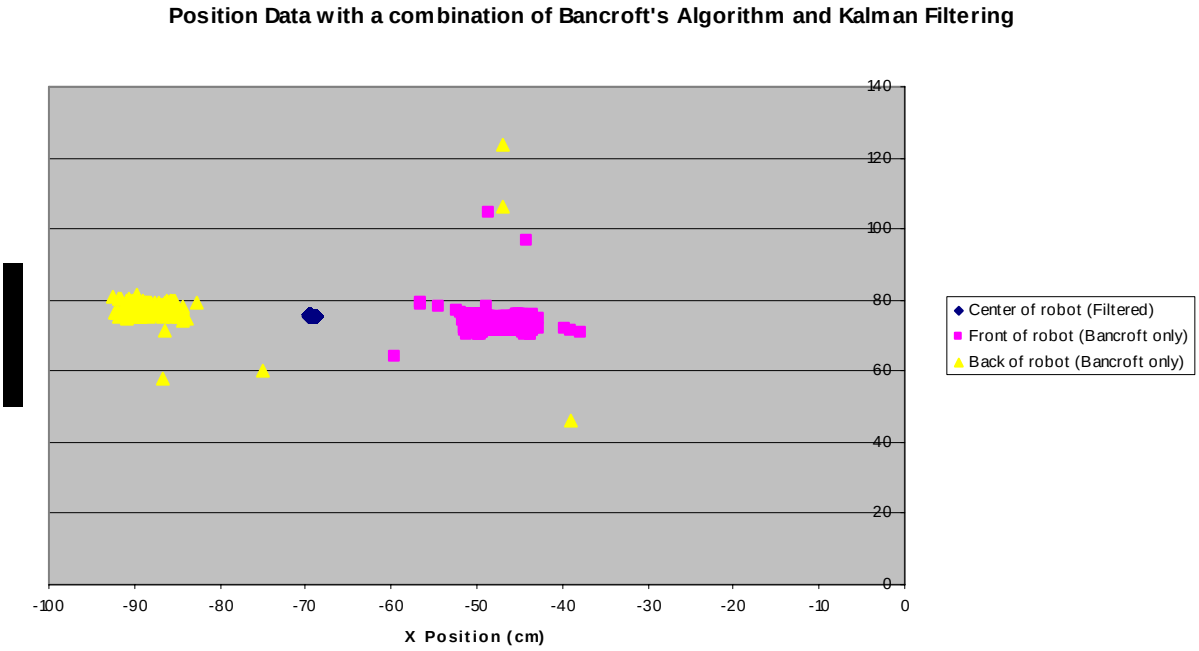

Figure VI. A plot of position data of a stationary robot using a Kalman filter with a combination of Bancroft's algorithm and odometry.

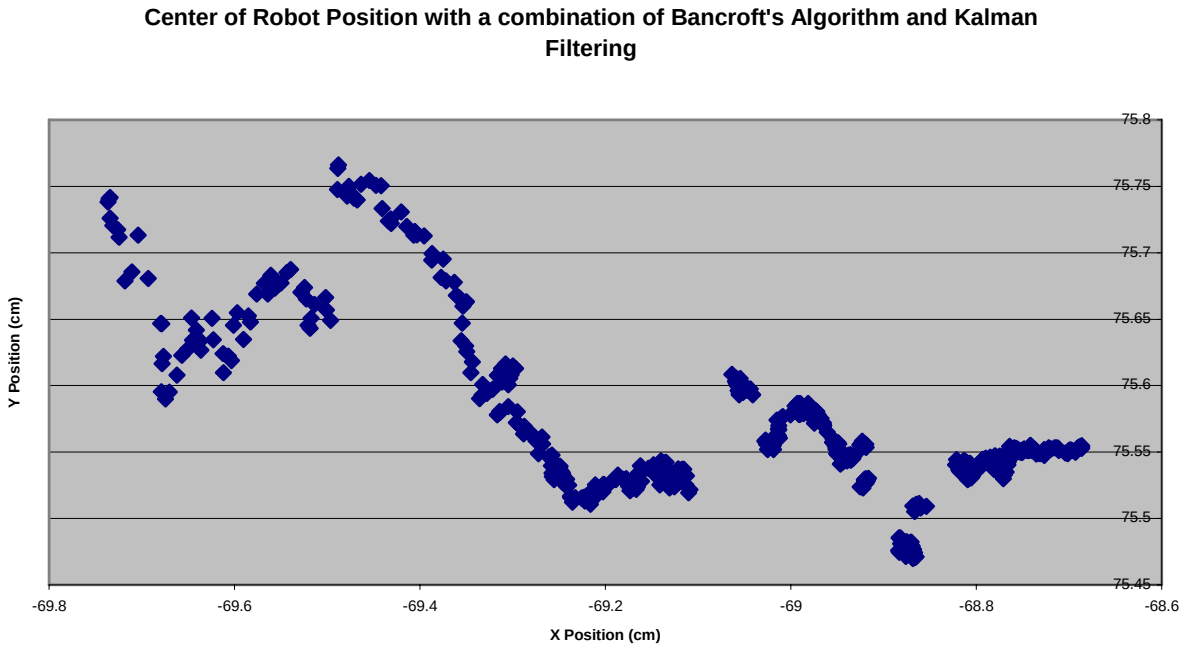

Figure VII. A close up plot of the Kalman filter position results.

While the combination of Bancroft's algorithm and the Kalman filter proved to be a viable choice for position estimate, the Kalman filter itself was capable of using the direct range

measurements. The next version of the Kalman filter that was tested attempted to eliminate the intermediary step, Bancroft's algorithm, to determine the absolute positions. The direct range measurements from the crickets were used as inputs into the Kalman filter instead and the results of the filter were expected to be more accurate because Bancroft's algorithm may have introduced additional error.

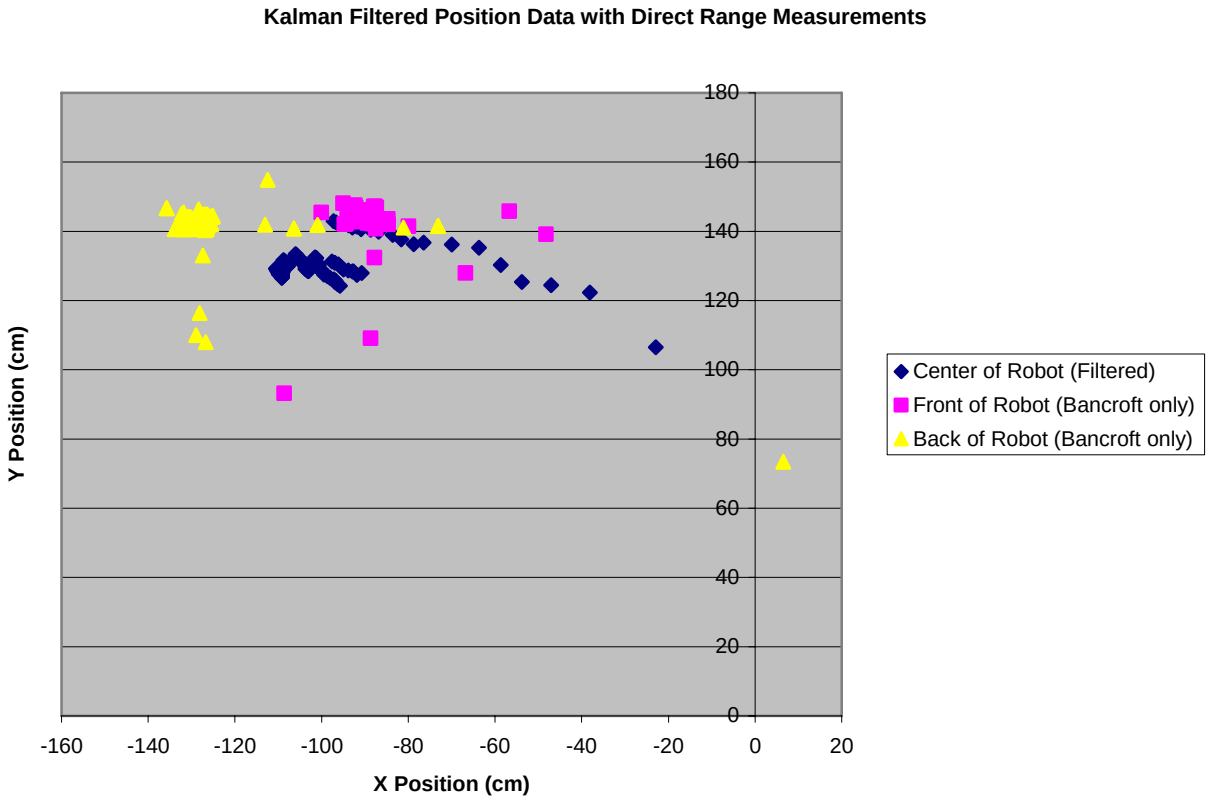

Figure VIII. A plot showing the results of the Kalman filter with direct range measurements and odometry data.

However, the direct range measurements had the disadvantage of being unable to determine a reasonable initial position. As one can see in Figure IV, the Kalman filter was unable to converge completely to the point in between the robot's Cricket units. Without being able to use Bancroft's algorithm or another similar algorithm, the initial position had to be set to some arbitrary point, in this case (0, 0), while the robot was actually at about (-110, 145). The final result of the Kalman filter with direct range measurements was impractical for use to carry out the intended pursuit strategies because of the lag between the initial position and an acceptable position estimate.

3.3 Motion Camouflage Pursuit

The final step of the project was to implement the pursuit strategies onto the robots with the improved position estimation techniques. As the Kalman filter with Bancroft and odometry data converged more quickly to the actual positions of the robots, it was more appropriate to use it as the input to the control laws. The resulting plots of the pursuer and evader were very smooth and accurate, with the exception of occasional jumps in the path due to poor Cricket data. However, these inaccuracies were quickly corrected by inputting additional Cricket data.

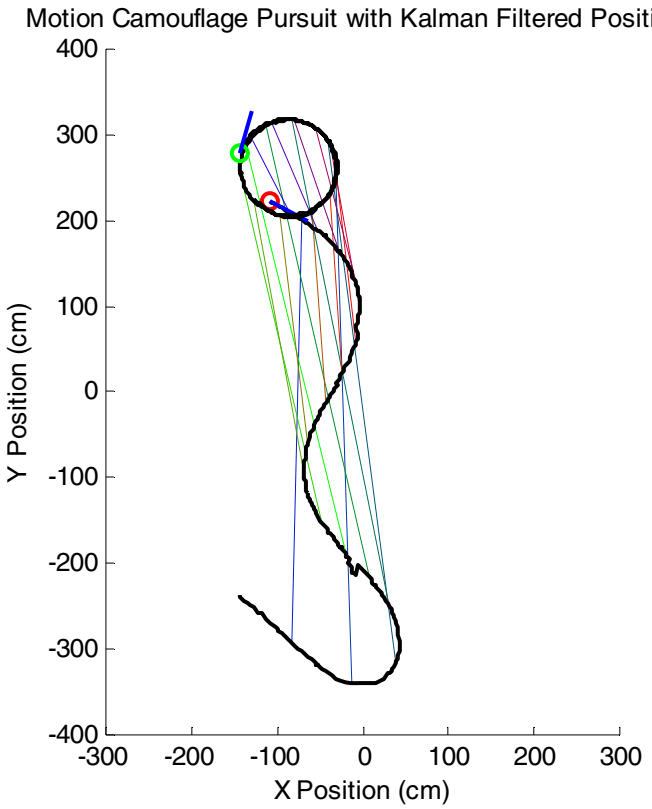

Figure IX. A plot of the positions of both the evader and the pursuer during a motion camouflage pursuit.

In Figure V, the pursuer, in red, is initially located on the bottom and the evader, in green, is located on the top. The evader was programmed to move at 4 centimeters per second in a circle and the pursuer was programmed to follow the evader at 5 centimeters per second. It can be seen in the figure that at each time instant, the pursuer was able to follow a path that kept it within the constraints of motion camouflage. To verify this, one can note that the baseline vector from the pursuer to the evader stays almost parallel. The drift of the line over time is because the control law for motion camouflage has no memory and steers based on the current state only. If

the baseline vector rotates at any instant, the motion camouflage control will guide it to maintain the new baseline vector angle.

4. Conclusion

Several conclusions were drawn after the completion of this project. First, there were a variety of problems with the Cricket units, including multipathing, measurements while moving, temperature gradients, and etc. These issues prevent Cricket, as a standalone device, from being able to effectively determine the position of a robot. To improve this model, experimentation was done with both versions of the extended Kalman filter. It was determined that the filter combining the more accurate, dead reckoning odometry data and the noisy, intermittent Cricket data was able to compensate for the deficiencies in the Cricket only model. In particular, the Bancroft/Kalman positioning strategy was able to converge quickly to within a couple centimeters of the actual position. The update frequency now depended on the onboard odometry device, which was several times faster, and not the Cricket device, which was used then for wheel slippage correction. Finally, using the Kalman filtered positions; it was possible to implement the motion camouflage control law on the robots. By feeding the control law position data for both robots, the pursuer was successfully able to capture the evader while maintaining a constant position with respect to a fixed frame.

5. References

1. Ghose, K., Horiuchi, T., Krishnaprasad, P. S. & Moss, C. 2006. Echolocating bats use a nearly time-optimal strategy to intercept prey. PLoS Biol. 4, 865–873, e108.
2. Haggag, H., Mehraei, G. 2006. Robot interaction using cricket, an indoor positioning system. MERIT program summer research paper.
3. Hristu-Varsakelis, D., Anderson, S., Zhang, F., Sodre, P., & Krishnaprasad, P.S. 2003. A Motion Description Language for Hybrid System Programming.
4. Justh, E. W., Krishnaprasad, P.S. 2006. Steering laws for motion camouflage. Proc. R. Soc. A FirstCite Early Online Publishing.
5. Justh, E. W., Krishnaprasad, P.S. 2002. A simple control law for UAV formation flying. Institute for Systems Research Technical Report TR 2002-38.

6. Lioi, J. 2006. Simulations of robotic pursuit using Matlab and Simulink. MERIT program summer research paper.
7. Priyantha, N., Chakraborty, A., Balakrishnan, H., 200. The Cricket Location-Support system, Proc. 6th ACM MOBICOM, Boston, MA, August.
8. Smith, A., Balakrishnan, H., Goraczko, M., Priyantha, N. June 2004. Tracking Moving Devices with the Cricket Location System, Proc. 2nd USENIX/ACM MOBISYS Conf., Boston, MA.
9. Watton, A., Kydon, D. W., 1969. Analytical aspects of the n-bug problem. American Journal of Physics, 37:220:221.
10. Welch, G., Bishop, G., 1995. "An Introduction to the Kalman Filter", Tech. Rep., TR 95-041, Dept of Comp. Sci. Uni. of North Carolina at Chapel Hill.