PRINCIPLES OF RADAR TARGET TRACKING

Stephen Chan, Brian Dai, Adam Lloyd, Jonathan MacMillan,
Alexander Morrison, Joshua Newman, Michael Sobin

Advisor: Mr. Randy Heuer
Assistant: Zachary Vogel

ABSTRACT

Tracking targets with radar is an important step in ensuring safety in such endeavors as air travel or military operations. To account for inherent inaccuracies in raw radar measurements of position, and to obtain accurate velocity data, we implemented an algorithm called the Kalman Filter. The resulting track data was used, in conjunction with algebraic and trigonometric methods, to simulate target interception and collision prevention. Our system led an interceptor aircraft to its target and warned pilots of a potential collision, proving the effectiveness of our filter.

INTRODUCTION

Goals of Radar Tracking

Radar is a tool which measures the position of targets. Radar tracking uses the information from the radar system along with an appropriate algorithm to determine the track of a target. Using the information generated by the radar system and the algorithm, the radar tracking system can be used for more sophisticated tasks such as sending an interceptor at any moving target as well as predicting and preventing a collision between two targets.

Complications from Radar Measurement

A radar system alone is insufficient for tracking targets. Radar can only measure the position of an object, not its velocity. There is also noise found in all radar measurements that leads to error in collected data. This noise comes in two forms, state noise and measurement noise. State noise occurs as a result of changes that affect the target's position and velocity, or state, such as pilot control inputs and weather conditions. The measurement noise is inherent in all radar systems and occurs when noise generated by the device or external sources interferes with the measurement [1].

KALMAN FILTER

The Kalman filter [1] is an efficient algorithm for estimating the state of a process that varies with time and has inherent Gaussian noise. It is an appropriate choice for the tracking problem for several reasons; it is computationally simple, estimates the error in its predictions, copes with measurements taken at inconsistent time intervals, and can be extended to track many targets simultaneously.

This algorithm was developed in the early 1960s by Rudolf Kalman to track spacecraft [2]. It includes in its calculations adjustments for both measurement noise and state noise, and is able to predict with increasing accuracy the state of a process.

Unlike some alternatives, the Kalman filter does not store all past measurements. Instead, it stores internal representations of error, which incorporate all previous measurements. This makes the filter simple and relatively easy to implement, allowing it to work in real time.

Model

The filter uses matrices to store all data, including state, error, and noise information. The algorithm estimates the time-varying state **x** and accounts for Gaussian noise. It also relates the obtained measurement to the true state of the object of measurement. Thus, the noisy process can be expressed as the following:

$$\mathbf{x}_{k+1} = \mathbf{\Phi}\mathbf{x}_k + \mathbf{q}_k \tag{1}$$

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{r}_k \tag{2}$$

Equation (1) shows that the true state $\mathbf{x}_{k+1}$ at a time step $k + 1$ is the sum of a transformation, multiplication by $\mathbf{\Phi}$, of the previous state $\mathbf{x}_k$ and state noise **q** that is an instance of normally distributed noise at that time step. Equation (2) expresses the measurement $\mathbf{y}_k$, such as that given by radar, as a function of the true state at time step k. The measurement is given by multiplying transformation matrix **H** by the state and adding an instance of normally distributed random noise, $\mathbf{r}_k$. The measurement does not exactly reflect the true state because of the noise, so the objective of the Kalman filter is to estimate the true state as closely as possible from all available noisy measurements. To do this, the algorithm uses the known state information to forecast the state at the time of the next measurement, then uses measurement data to correct this prediction. The algorithm's improvement in accuracy over raw measurements comes from judging whether the prediction or measurements should be weighted more heavily in determining the new state.

Predict

In this first phase of the algorithm, no measurement data is used. Instead, given a time step specifying the time difference between the last measurement and the next measurement, predictions of the new state are made in equation (3) and predictions of the new state covariance, **P**, are made in equation (4). Error matrix **Q** specifies the known covariance of the state noise, which defines the greatest accuracy with which the state may be known.

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{\Phi}\hat{\mathbf{x}}_k \tag{3}$$

$$\mathbf{P}_{k+1|k} = \mathbf{\Phi}\mathbf{P}_k\mathbf{\Phi}^T + \mathbf{Q} \tag{4}$$

The notation $\mathbf{x}_{k+1|k}$ represents the value of **x** at time step $k + 1$ when data up to time step k is known. Thus, the equation (3) predicts the state and equation (4) predicts the state covariance at time step $k + 1$.

Correct

After the prediction is made, new measurement data stored in $\mathbf{y}_k$ is utilized. To properly estimate the measurement error here, the measurement covariance matrix **R** contains variances and covariances of the measurement data. The Kalman gain matrix **K** an intermediate matrix and is the means by which the weighting is performed. The value of this matrix is used to correct the value of the state and state covariance matrices.

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T\left[\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R}\right]^{-1} \tag{5}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left[\mathbf{y}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}\right] \tag{6}$$

$$\mathbf{P}_{k|k} = \left[\mathbf{I} - \mathbf{K}_k\mathbf{H}\right]\mathbf{P}_{k|k-1} \tag{7}$$

Because new data is being used to refine the state prediction, the notation used is slightly different: *k* refers to the time step being currently updated, and *k – 1* represents the previous time step (in the predict phase, these were *k + 1* and *k*, respectively). Equation (5) calculates the Kalman gain, the weighting between prediction and measurement. This is used in equation (6), which actually corrects the predicted state. Finally, the equation (7) corrects the state covariance matrix.

Iteration of these prediction and correction steps as new measurement data become available provides an efficient solution for the problem of radar tracking.

IMPLEMENTATION OF FILTER

Our implementation of the Kalman filter was written in the Microsoft's Visual Basic .NET programming language. We had available a matrix library, Matlib [3], capable of handling all necessary matrix operations. The value of the state **x** was chosen to contain position and velocity, quantities important to the tasks we planned on completing, as shown in equation (8).

$$\mathbf{x} = \begin{bmatrix} x\text{ - position} \\ x\text{ - velocity} \\ y\text{ - position} \\ y\text{ - velocity} \end{bmatrix} = \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix} \tag{8}$$

To transform one state into the next, a linear transformation was defined. Velocity and time step, are translated into a change in position. For a time step of length Δt, the state transformation matrix is defined in equation (9).

$$\mathbf{\Phi} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

As described previously, radar measurements provide only position, not velocity. The input to the filter consists of measured position data in the form shown in equation (10).

$$\mathbf{y} = \begin{bmatrix} \text{measured } x\text{-position} \\ \text{measured } y\text{-position} \end{bmatrix} = \begin{bmatrix} x_m \\ y_m \end{bmatrix} \tag{10}$$

The corresponding transformation matrix **H**, as determined from the model's equations must be as defined in equation (11).

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{11}$$

The error matrix **Q** contains variances and covariances of state error, representing the process noise inherent in the radar tracking process. The elements are specified in equation (12). This matrix represents the error statistics of state noise **q** from the Kalman filter model.

$$\mathbf{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_x\sigma_{v_x} & \sigma_x\sigma_y & \sigma_x\sigma_{v_y} \\ \sigma_x\sigma_{v_x} & \sigma_{v_x}^2 & \sigma_{v_x}\sigma_y & \sigma_{v_x}\sigma_{v_y} \\ \sigma_x\sigma_y & \sigma_{v_x}\sigma_y & \sigma_y^2 & \sigma_y\sigma_{v_y} \\ \sigma_x\sigma_{v_y} & \sigma_{v_x}\sigma_{v_y} & \sigma_y\sigma_{v_y} & \sigma_{v_y}^2 \end{bmatrix} \tag{12}$$

Lastly, the error matrix **R**, describing the measurement error, is defined in equation (13). This corresponds to the measurement error term **r** in the model.

$$\mathbf{R} = \begin{bmatrix} \sigma_{x_m}^2 & \sigma_{x_m}\sigma_{y_m} \\ \sigma_{x_m}\sigma_{y_m} & \sigma_{y_m}^2 \end{bmatrix} \tag{13}$$

To use the algorithm, the state **x** must be initialized. The position terms may be read directly from radar measurements, but radar does not measure velocity so the velocity must be calculated. A simple and effective initialization is to use the average velocity between the first two available measured positions. Call the first two position measurements (x_0, y_0) and (x_1, y_1), separated by time difference Δt. The initialization of **x** is as follows in equation (14).

$$\mathbf{x}_0 = \begin{bmatrix} x_2 \\ (x_1 - x_0)/\Delta t \\ y_2 \\ (y_1 - y_0)/\Delta t \end{bmatrix} \tag{14}$$

BASIC SCENARIOS

2-D Cartesian Coordinates

The first scenario for target tracking with the Kalman filter involved data provided in Cartesian form. This basic data form allowed for easy implementation since the coordinate system the filter uses was the same as the one in which the data were measured. Though radar data of this form were not realistic, the form was conducive to initial testing of filter code.

After initializing the filter, we ran the program to test if it worked. Our program yielded favorable results as the distance from our prediction to the actual location of the object diminished. This difference, called the residual, is shown in equation (15), where $\hat{x}$ and $\hat{y}$ represent the estimated values of x and y predicted by the filter at a given time and x and y represent the actual position of the target at that time. The residual of the measured data can be found using equation (15) with the measured positions of x and y used for $\hat{x}$ and $\hat{y}$.

$$residual = \sqrt{(\hat{x} - x)^2 + (\hat{y} - y)^2} \tag{15}$$

After a small number of radar measurements, the residuals for our prediction became less than those of the measured data (Fig. 1). This meant that there was less error in our predictions than in the radar measurements. As time progressed, the accuracy of the filter increased due to an increased number of measurements. We had successfully compensated for the state and measurement noises in our program.

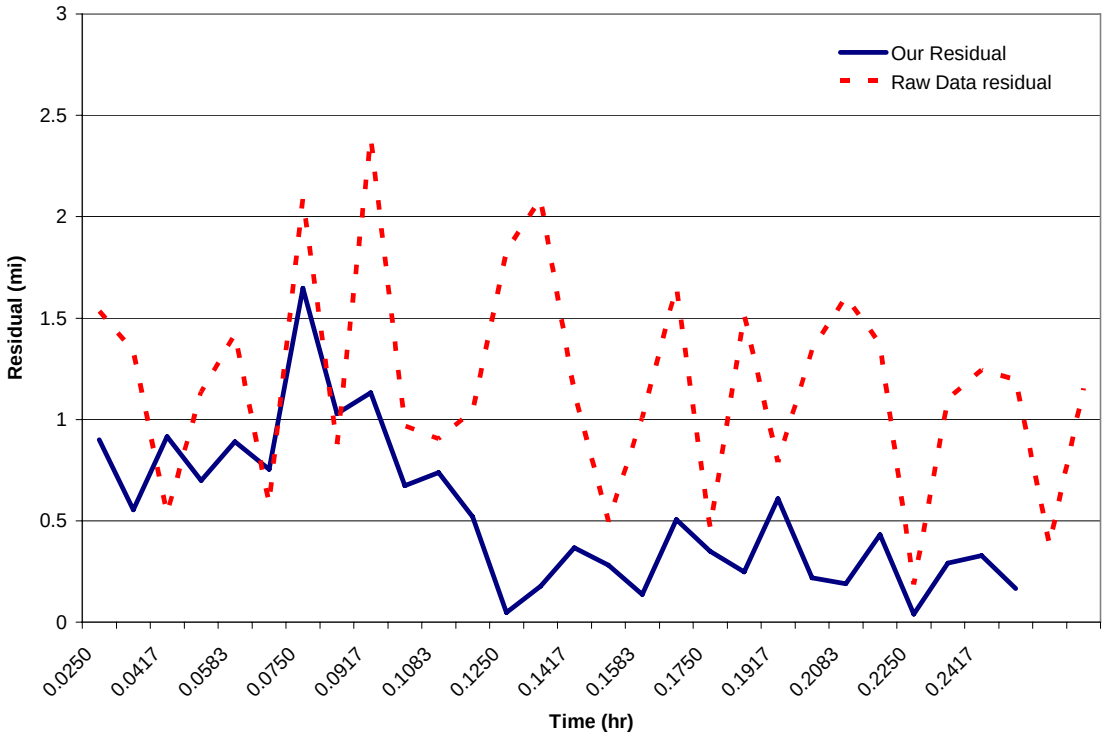

Fig. 1: Residuals for Cartesian Coordinates

Range and Bearing

Transforming Coordinates

In the previous scenario we showed that our implementation of the Kalman filter works, but in real life data is never measured by radar in Cartesian coordinates. Radar systems report the location of an object in range and bearing. The range is the distance the object is from the sensor and the bearing is the angle measured from North to the object (Fig. 2). Converting these figures to Cartesian coordinates is necessary before running the filter because our filter is designed to use only Cartesian inputs.

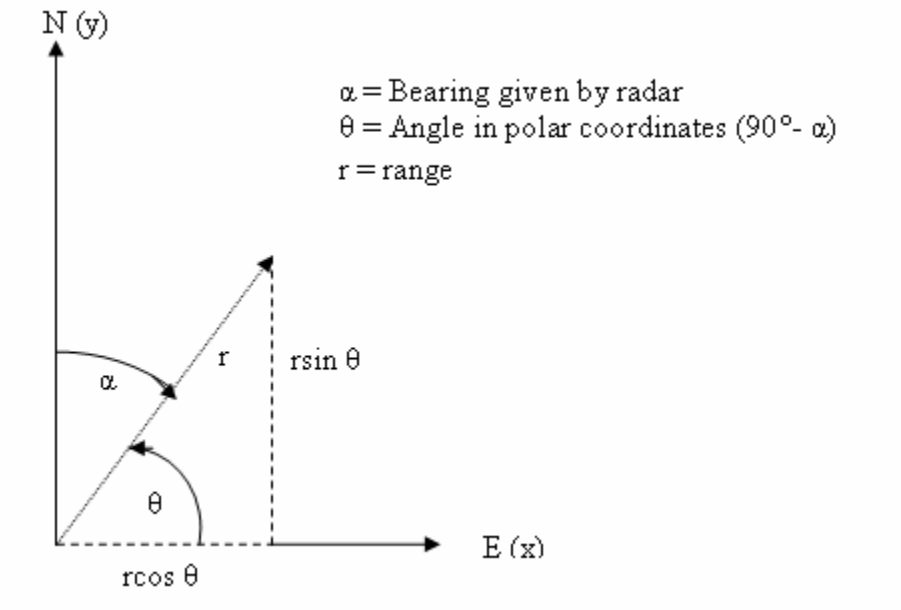

Fig. 2: Conversion from Range and Bearing to Cartesian

The first step in the conversion is to convert the range and bearing measurement to one that the filter can use. The computer computes angles based on polar coordinate systems rather than the range and bearing system. To convert, we simply use equation (16). Using this value for the angle, we can now convert rather easily to Cartesian coordinates using equations (17) and (18), where r is the range and θ is the polar angle. These conversions are displayed in Fig. 2.

$$\theta = 90 - \alpha \tag{16}$$

$$x = r\cos\theta \tag{17}$$

$$y = r\sin\theta \tag{18}$$

These coordinates can then be used in the filter as they were in the previous example.

Transforming Error

Converting the range and bearing to Cartesian coordinates does not completely solve the dilemma posed by the range and bearing measurement. The measurement error must also be adjusted. This will affect the measurement covariance matrix, **R**, used in the calculations. With Cartesian measurements, the error of the measurements traces an ellipse with major and minor axes parallel to the x and y axes around the point. In our previous scenario the *x* and *y* errors were the same and so they defined an circle. When polar coordinates are used, the axes of this ellipse are no longer parallel to the *x* and *y* axes (Fig. 3).

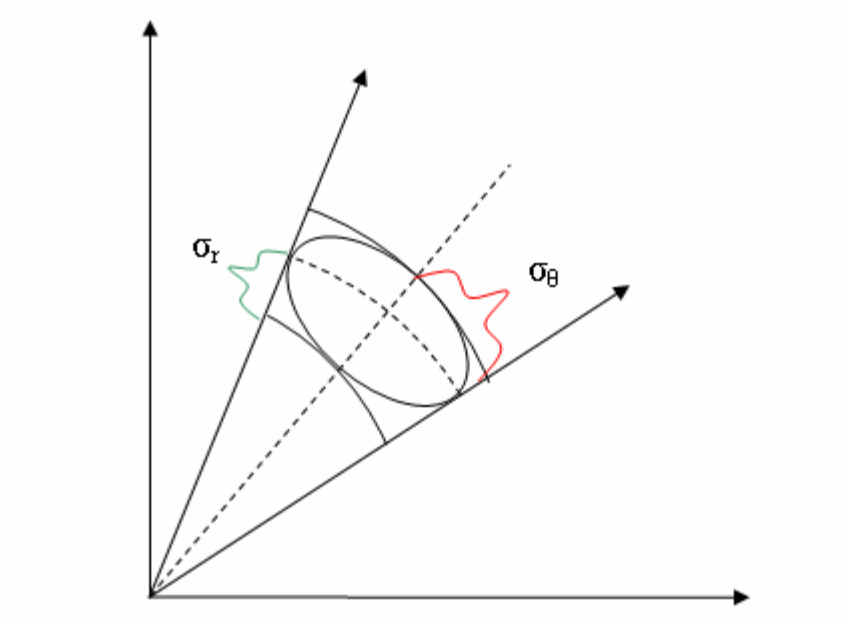

Fig. 3: Conversion of Range and Bearing Error to Cartesian

To convert the measurement errors of range and bearing into the covariance matrix we must use a series of transformations. In equations (19), (20), and (21), σ_x^2 and σ_y^2 are the variances of the *x* and *y* measurements and σ_{xy}^2 is the covariance of *x* with *y*. These are the same terms found in the **R** matrix defined in equation (13). The terms σ_r^2, and σ_θ^2 are the variances of the range and bearing measurements respectively, θ is the measured bearing, and *R* is the measured range [4].

$$\sigma_x^2 = \sigma_r^2 \cos^2\theta + R^2 \sin^2\theta\sigma_\theta^2 \tag{19}$$

$$\sigma_y^2 = \sigma_r^2 \sin^2\theta + R^2 \cos^2\theta\sigma_\theta^2 \tag{20}$$

$$\sigma_{xy}^2 = \frac{1}{2}\sin 2\theta\left[\sigma_r^2 - R^2\sigma_\theta^2\right] \tag{21}$$

These transformations define the new measurement covariance matrix, which is a linear representation of the true uncertainty. Whereas when receiving data in Cartesian coordinates the

covariance of x and y, σ^2_{xy}, was zero, it now takes on a value due to their dependence on each other.

Our Results

Using the transformations to Cartesian coordinates of the measurements and measurement errors, our tracking filter produced the residuals seen in Fig. 4. Our residuals are better than those of the raw data, showing that our filter can effectively track the target even though the radar measures range and bearing.

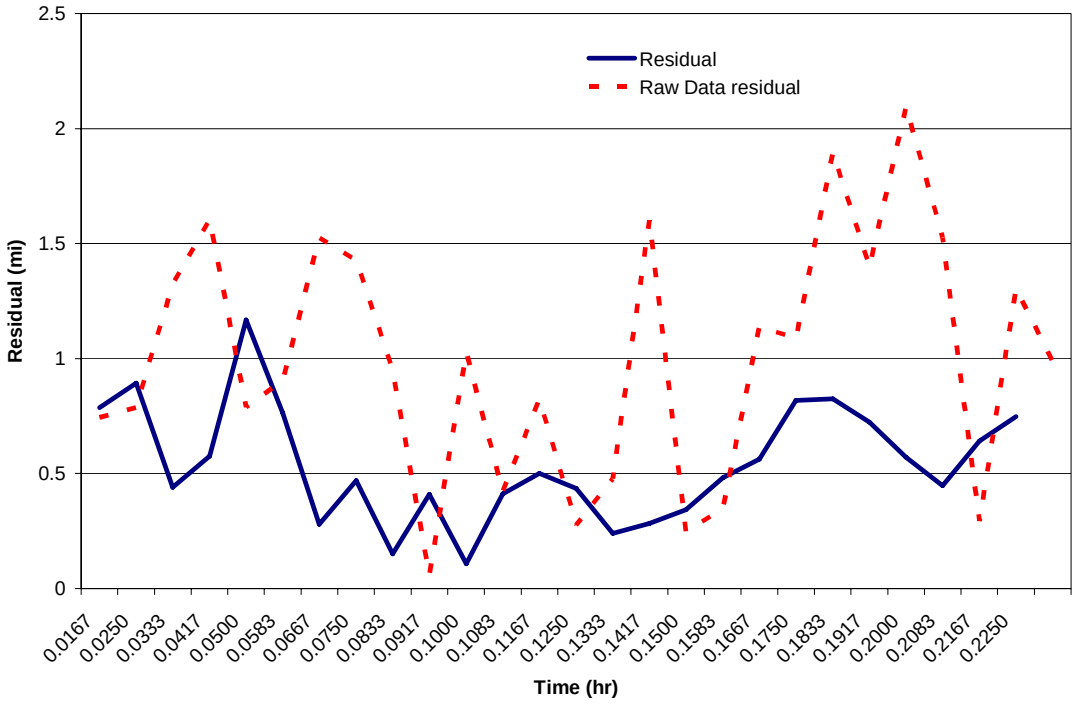

Fig. 4: Residuals of Range and Bearing

Multiple Radar Tracking

The third case introduced an additional element of complexity to the filter. Building on the information input of the past two cases, we now get out information from two different radar systems, neither located at the origin in the Cartesian plane. To use the data from the radars, the position readings had to be translated. By adding the location of the radar system to each data point the radar produced we are able to translate the data into a form that the filter can use.

The reason that multiple radars are used is because they can produce more accurate predictions. When two radar systems view an object, the error ellipse becomes the intersection of the two ellipses (Fig. 5).

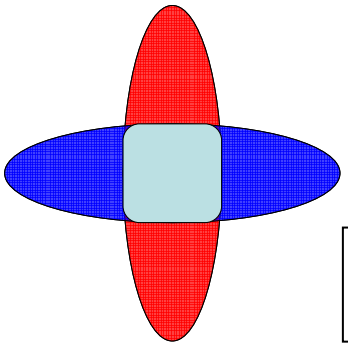

Fig. 5: Reduced Error From Multiple Radars

With a smaller error ellipse, the residuals drop considerably (Fig. 6). The second radar system is introduced into the problem at timestep 0.1083 hours. At this time, a spike in error is seen on the residual plot due to a difference in angle. A few timesteps afterwards when the second radar is working along with the first one, the average residual becomes much smaller, as shown in Fig. 6.

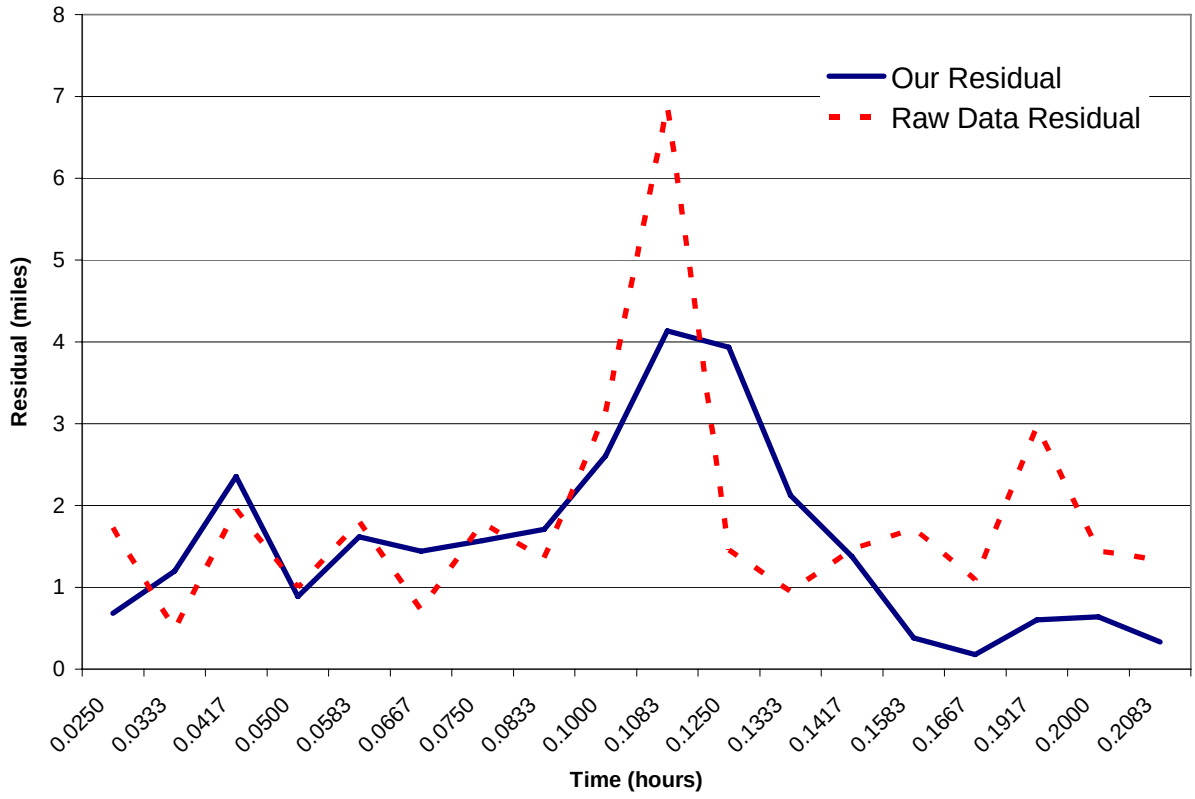

Fig. 6: Residuals from Multiple Radars

Tracking Multiple Targets

In real radar systems, there is rarely one object in the sky. Almost all tracking systems handle multiple targets, so there must be a method of doing this. When used correctly, multiple instances of the Kalman filter can actually be used to follow track targets, and its efficiency and simplicity allow it to do this without requiring sophisticated computing equipment.

To track multiple targets, the radar system must run the algorithm separately for each target. This approach considers each target independently, using the data for each target solely to update that target's state. Data associated with the algorithm is separate for target, as well. Thus, the problem lends itself well to an object-oriented approach (Fig. 7).

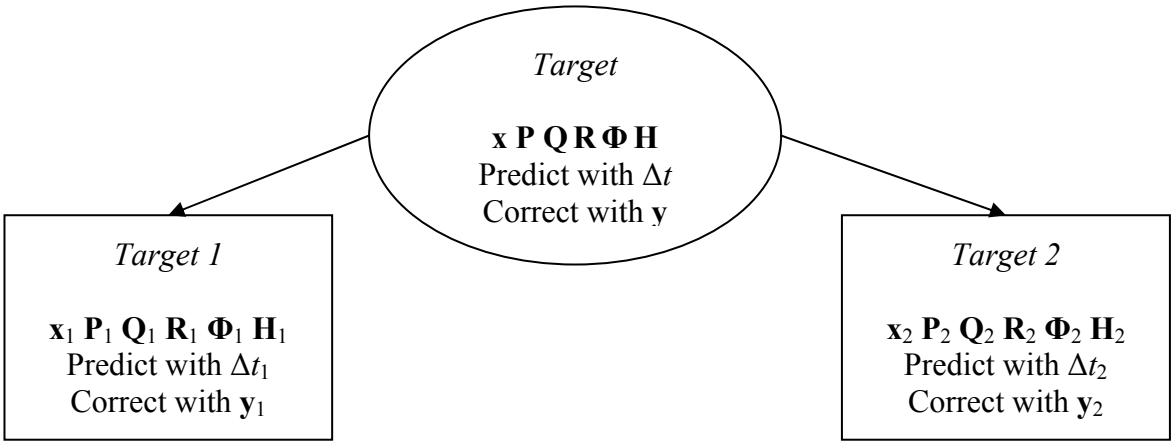

Fig. 7: Object-Oriented Design

After implementing a system for two targets with two instances of the Kalman filter tracking algorithm, we found the results were as expected: the filter correctly tracked each object.

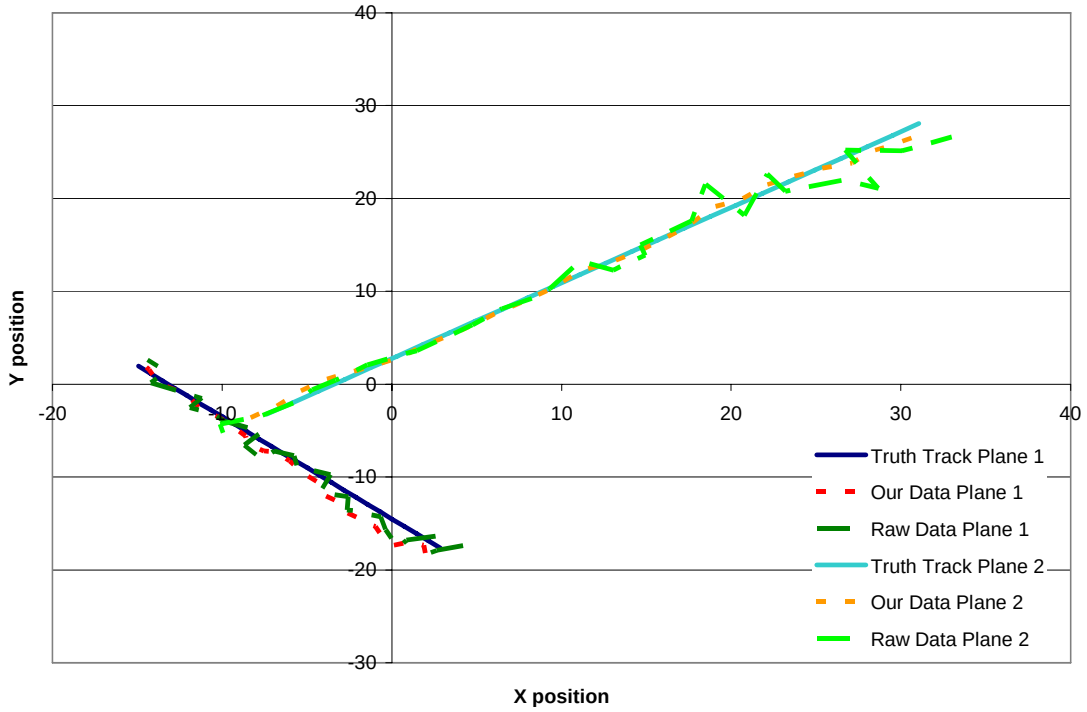

Fig. 8: Tracks of Multiple Targets

When compared to the truth data (Fig. 8), to which random noise was added to generate the simulation's measurements, the tracking algorithm's output more closely followed the true track of the target, proving the effectiveness of the implementation.

PRACTICAL USES OF THE FILTER

Collision Prevention

One very important use of radar in air-traffic control is to prevent aircraft collisions. Of course, warning of a potential collision must be provided in advance, to ensure that pilots have time to correct the situation. Considering any distance less than one mile to another target to be a dangerous range when aircraft are at the same altitude, the collision prevention algorithm must warn pilots when they are potentially in danger of colliding with another target in the future.

Algorithm

The algorithm will check for a potential future collision whenever two targets are within twelve miles of each other, as shown in Fig. 9. If a collision is predicted, the algorithm will warn the pilots at that point, far in advance.

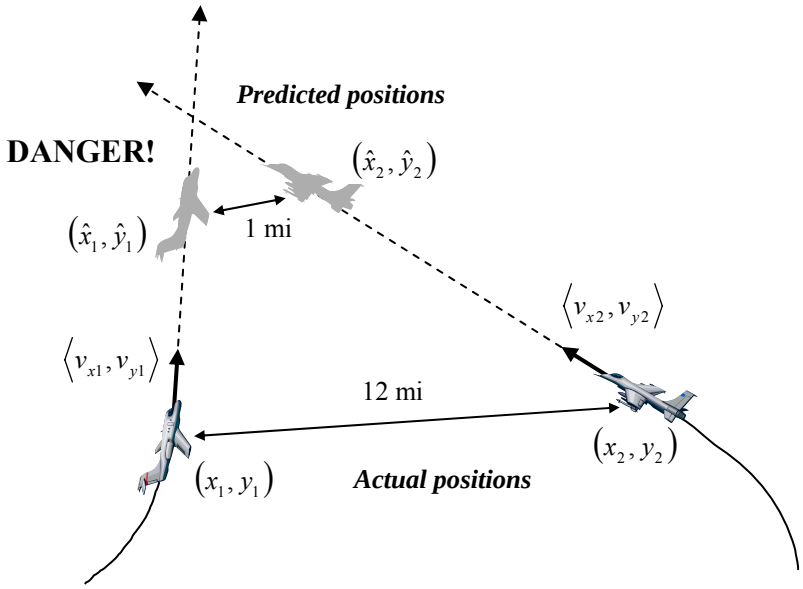

Fig. 9: Extrapolation in Collision Prevention

To perform this check, first define $t = 0$ to be the present time. Next, express the position of the plane at any time t in the future as a function of current positions (x_1, y_1), (x_2, y_2) and velocities $\langle v_{x1}, v_{y1} \rangle$ and $\langle v_{x2}, v_{y2} \rangle$ of target 1 and target 2, respectively, as in equations (22), (23), (24), and (25).

$$\hat{x}_1 = x_1 + v_{x1}t \tag{22}$$
$$\hat{y}_1 = y_1 + v_{y1}t \tag{23}$$
$$\hat{x}_2 = x_2 + v_{x2}t \tag{24}$$
$$\hat{y}_2 = y_2 + v_{y2}t \tag{25}$$

The pilots must be warned when the distance between these planes is less than one mile, a condition expressed in inequality (26):

$$\sqrt{\left(\hat{x}_2 - \hat{x}_1\right)^2 + \left(\hat{y}_2 - \hat{y}_1\right)^2} < 1 \tag{26}$$

Substitute for the predicted positions, and define the following quantities, equations (27), (28), (29), and (30), to simplify the inequality (31):

$$\Delta x = x_2 - x_1 \tag{27}$$
$$\Delta y = y_2 - y_1 \tag{28}$$
$$\Delta v_x = v_{x2} - v_{x1} \tag{29}$$
$$\Delta v_y = v_{y2} - v_{y1} \tag{30}$$

A quadratic inequality (32) in t is obtained:

$$\left(\Delta x + \Delta v_x t\right)^2 + \left(\Delta y + \Delta v_y t\right)^2 < 1 \tag{31}$$
$$\left(\left(\Delta v_x\right)^2 + \left(\Delta v_y\right)^2\right) t^2 + 2\left(\Delta x \Delta v_x + \Delta y \Delta v_y\right) t + \left(\left(\Delta x\right)^2 + \left(\Delta y\right)^2 - 1\right) < 0 \tag{32}$$

The inequality can be solved by finding the zeroes of the quadratic function on the left side of inequality (32). The coefficients to be substituted into the quadratic formula to find these zeroes are specified in equations (33), (34), and (35).

$$a = \left(\Delta v_x\right)^2 + \left(\Delta v_y\right)^2 \tag{33}$$
$$b = 2\left(\Delta x \Delta v_x + \Delta y \Delta v_y\right) \tag{34}$$
$$c = \left(\Delta x\right)^2 + \left(\Delta y\right)^2 - 1 \tag{35}$$

The solution, if it exists, is a set of real number values of t that satisfy in the inequality. If there is no solution or if both endpoints of the interval are negative, there is no time t in the future in which the planes will be within one mile of each other. If any subinterval within this interval is positive, which means there is a solution, the planes will be within one mile of each other in that interval and the pilots should be warned of the danger.

Testing

We tested this algorithm on sample data in which the planes were on a course that would take them within one mile of each other. A section of the resulting tracks is shown in Fig. 10.

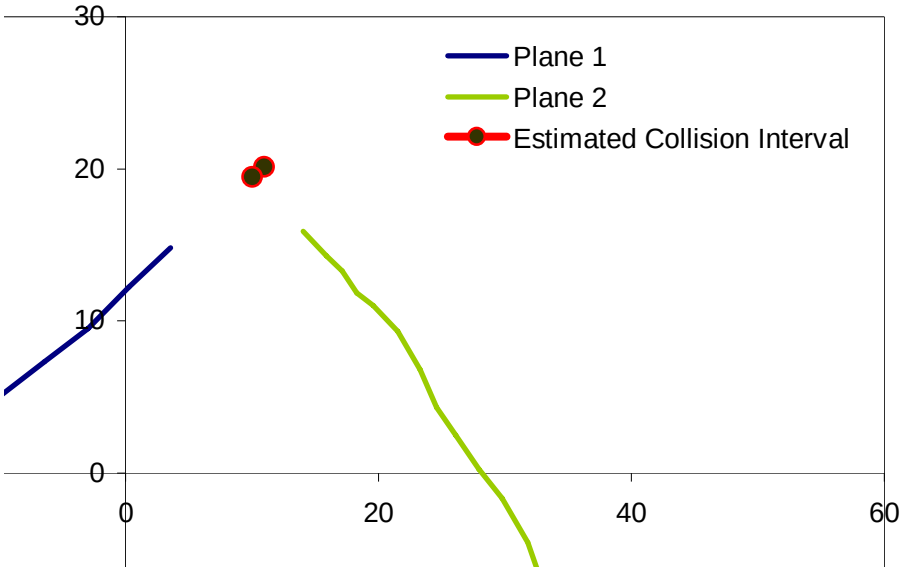

Fig. 10: Collision Prevention Results

The two tracks are the paths of the planes according to the Kalman filter. Each is evaluated independently until they are within twelve miles of each other. At this point, the algorithm detects that the planes will be within one mile of each other at some point in the future and warns the pilots. The positions at the time of warning are the last ones shown on the graph.

Two additional points from the future track of plane 1 are shown to provide some idea of the predicted collision's location. The solution to the inequality was an open interval whose endpoints, in this case, were both positive. Thus, we had two future times between which the planes would be within one mile of each other. The first point not on a track is the position of plane 1 at beginning of this time interval and the second point is the position of plane 1 at the end. However, the filter's warning was provided in advance, so the pilots would have time to avoid the potentially dangerous situation.

Maneuvering Targets

Up until this point, all of the targets we have tracked were moving in straight lines. In real life this is not always the case. The filter must be able to account for maneuvering targets. Since our implementation of the Kalman filter is designed for linear motion, we will treat our target track as a series of linear tracks. In order to track these maneuvering targets, the filter must be able to recognize when the target object has changed its course and then reinitialize itself based on the target's new track.

Determining When the Target has Changed Course

To determine if the object has changed course, we must determine whether or not it lies in the predicted error ellipse for the measured point. This ellipse is formed by the positional components of the state covariance matrix and described in equation (25), where $n\sigma$ represents

the amount of measurement variance accounted for, $\mathbf{R}_{xx}^{-1}$ is the inverse of σ_x^2, $\mathbf{R}_{xy}^{-1}$ is the inverse of $\sigma_x\sigma_y$, and $\mathbf{R}_{yy}^{-1}$ is the inverse of σ_y^2. These σ terms come from the **P** matrix described in equation (4). Term $\hat{x}$ is the predicted *x* position, $\hat{y}$ is the predicted *y* position, and *x* and *y* are any *x* and *y* values [5].

$$(n\sigma)^2 = (x-\hat{x})^2 R_{xx}^{-1} + 2(x-\hat{x})(y-\hat{y})R_{xy}^{-1} + (y-\hat{y})^2 R_{yy}^{-1} \tag{36}$$

If we allow for $n = 1$, there a 46% chance that a measured data point will fall inside the given ellipse if the target has not maneuverd. To increase the probability that a measured point falls inside the ellipse we must allow for more deviation from the predicted value. We set $n = 3$, there is a 98% probability that a measured point lies within the ellipse if the target has not maneuvered.

The next step is to determine if the object has changed its course. The first step is to check whether it lies within the 3σ ellipse. Inequality (37), where $\hat{x}$, $\hat{y}$, $\mathbf{R}_{xx}^{-1}$, $\mathbf{R}_{xy}^{-1}$, and $\mathbf{R}_{yy}^{-1}$ are the same as in inequality (36), x_m is the measured x position, and y_m is the measured y position, will only be true if the measured point lies within this 3σ ellipse.

$$(x_m-\hat{x})^2 R_{xx}^{-1} + 2(x_m-\hat{x})(y_m-\hat{y})R_{xy}^{-1} + (y_m-\hat{y})^2 R_{yy}^{-1} \le (n\sigma)^2 \tag{37}$$

If the point does not satisfy this inequality it lies outside the designated ellipse and therefore tells the filter the target might have maneuvered. We cannot be certain the path has changed yet because only 98% of all possible measurements lie within the ellipse. In our implementation of the filter, three consecutive points that lie outside the state covariance ellipse will trigger a reinitialization of the filter based on the targets new path. The probability that 3 consecutive measurements will lie outside the ellipse is .4%, so we can be fairly certain the target has changed its course.

Reinitializing the Filter

Once the filter has determined that the target has changed its course, it must reinitialize based on the new path. One part of the process is reinitializing the state matrix, **x**. When the target maneuvers, the state matrix, **x**, is no longer accurate. Thus the *x*, *y*, v_x and v_y values must be reset via the same methods used to originally initialize them. To determine an initial velocity, we use equation (14) again with the terms x_0, y_0, y_1 and x_1 replaced by the values of the last two recorded data points. These points were both outside the 3σ ellipse and thus represent the target's new state. The new position of the target is then determined by the last collected data point.

The other part of the reinitialization process is changing **P**, the state covariance matrix. As the filter runs, the components of **P** become smaller and approach the state noise covariance matrix, **Q**. This causes the Kalman gain matrix to put more weight on the predicted value than on the measured value. Once we have determined the target's course has changed, the weight must once more be placed heavily on the measured data as opposed to the predictions. To do this, we set **P** equal to its value after the original initialization. The variance of the terms in **P** must be set

very high again due to the dearth of data used to determine the path. With **P** reinitialized, the filter is now set to run based on the target's new state.

Our Results

Using the processes outlined above, we were able to implement a program that can take into account the target's change in direction. In Fig. 11, the path of the maneuvering target is graphed along with our predicted locations of it.

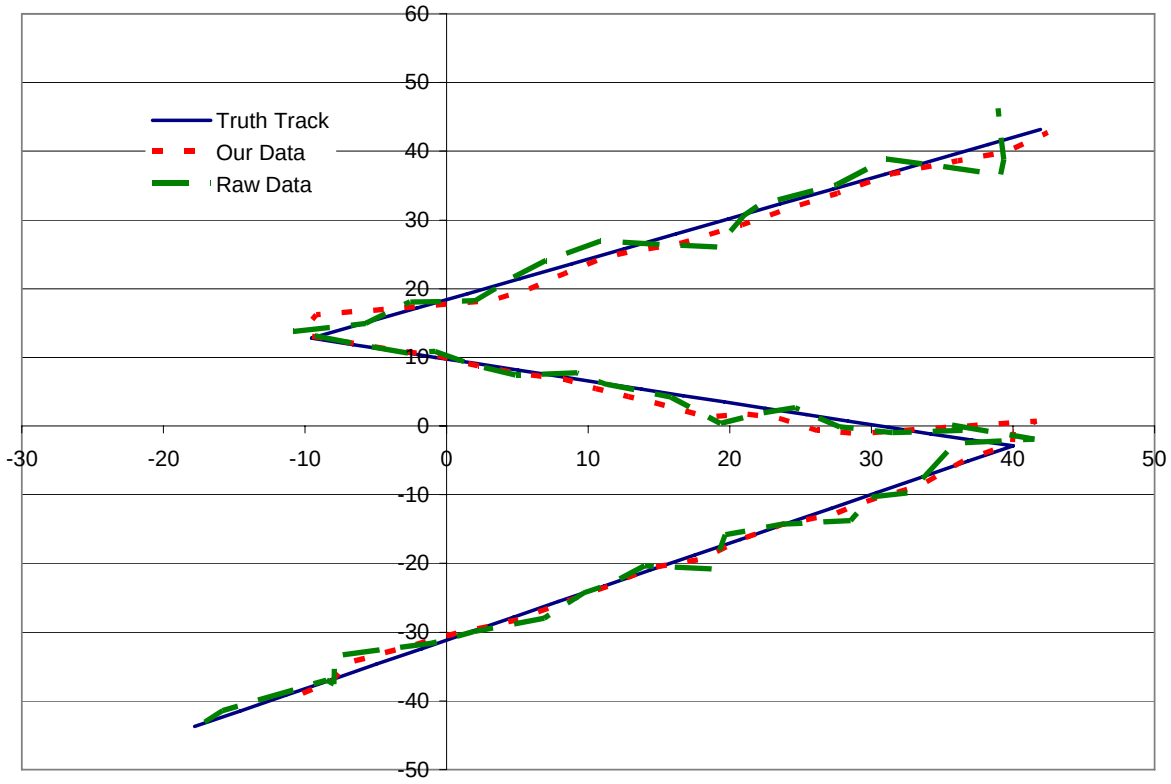

Fig. 11: Maneuvering Plane Track

As the target turns, the predicted position will continue in the original direction for a few data points before turning while the filter is initialized. Fig. 12 shows the graph of the residuals of our data. The large spikes occur when the target changes course because the filter has not reinitialized. A few data points later the filter is once more tracking the target with better accuracy than the radar measurements alone.

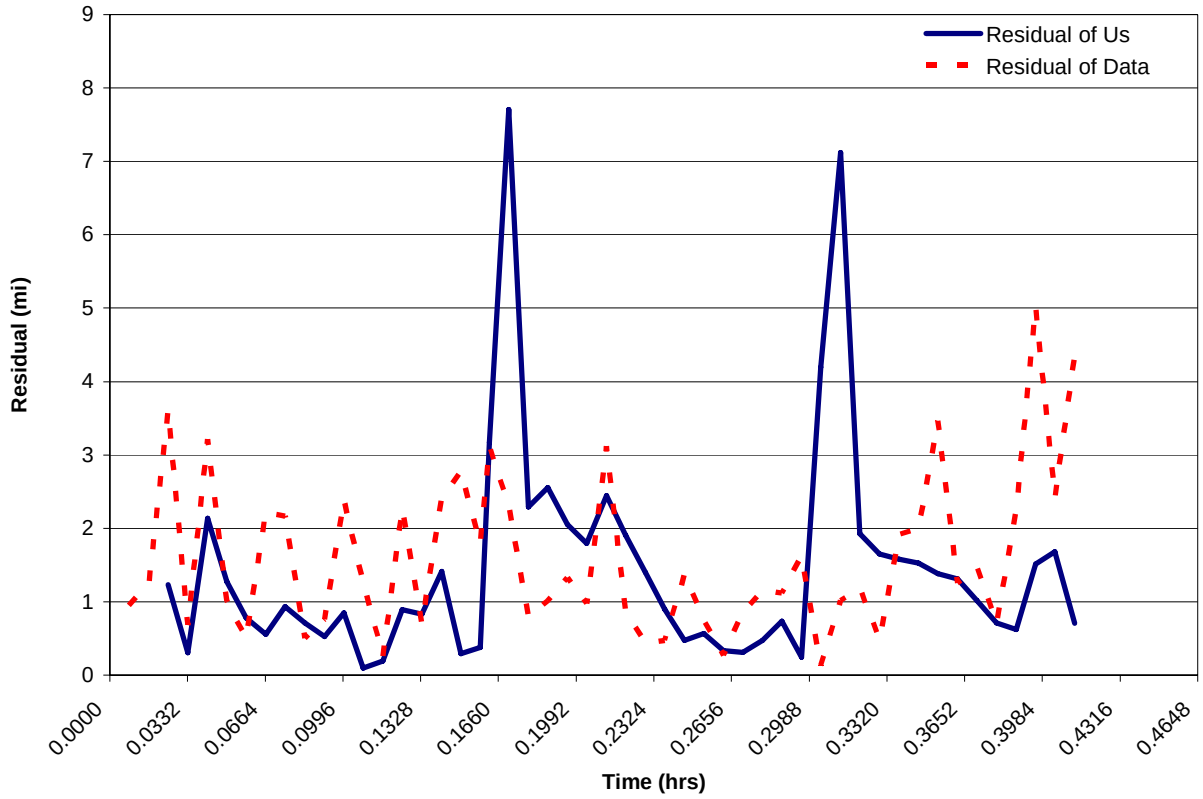

Fig. 12: Maneuvering Plane Residuals

Intercepting a Target

The Process

Using the system we developed for maneuvering targets, we now must intercept one of these maneuvering targets. We will send an interceptor that travels at a constant speed, v_I, to intercept the target. At each point in time we will instruct it to fly the shortest route to the target's position at the time of predicted interception. To do this we will give the interceptor an angle from the *x* axis, γ (Fig. 13), at which it must travel to intercept the target each timestep.

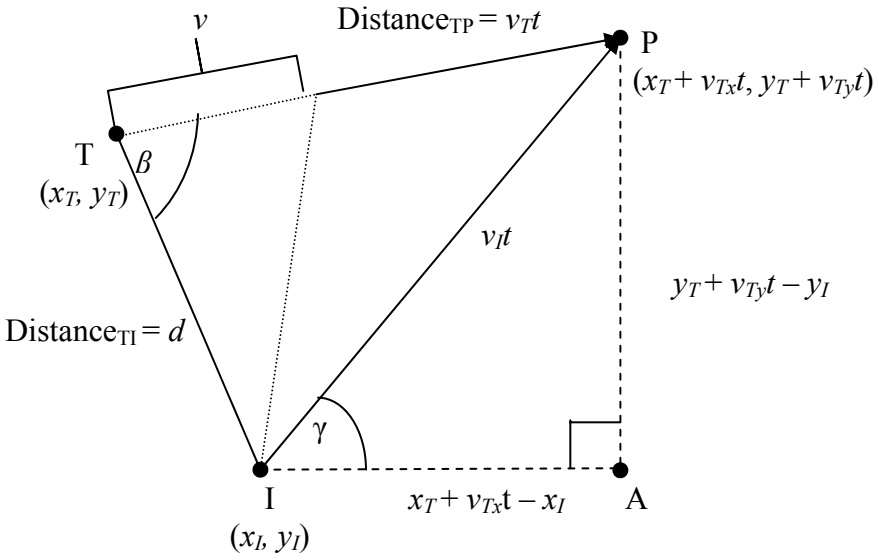

Fig. 13: Interception Calculations

Fig. 13, where v_T is the speed of the target, v_I is the speed of the interceptor, v_{Tx} is the x velocity of the target, and v_{Ty} is the y velocity of the target, shows the position of the target, T, at the current time, the position of the interceptor, I, at the current time, and the projected point of intersection, P. These three points trace out the triangle shown by the darker lines. The angle β in this triangle is independent of what time, t, the intersection occurs. For example, the triangle determined by the lighter lines shows a triangle formed at time $t = 1$ and the angle β is the same here as it is at any time t. Since the angle has not changed, we can now solve for β by setting $t = 1$. This yields the following equations where d, defined in equation (38), represents the distance from the interceptor to the target. Using them we can solve for the angle β, the value of which is given in equation (39)(40).

$$d = \sqrt{(x_I - x_T)^2 + (y_I - y_T)^2} \tag{38}$$

$$v_I^{\,2} = v_T^{\,2} + d^2 - 2v_T d\cos\beta \tag{39}$$

$$\beta = \arccos\frac{v_I^{\,2} - v_T^{\,2} - d^2}{-2v_T(d)} \tag{40}$$

Since all the variables in equation (40) are known for any time t, β is valid for all times t. The next step is to solve for the time t when the interceptor will intersect the target. Once more using the Law of Cosines to get equation (41) we can derive the quadratic equation (42), for t. Using the formula for the roots of a quadratic equation to get equation (43), we are able to solve for two times where the interceptor can intersect the target. If both of these times are positive, we choose the smaller of the two to expedite the interception. If one is positive and one is negative, it means the interceptor could have intercepted the target at a previous time but did not and so now has only one possible course of action, which is determined by the positive time.

$$(v_I t)^2 = (v_T t)^2 + d^2 - 2v_T td\cos\beta \tag{41}$$

$$v_T{}^2 t^2 - 2v_T td\cos\beta + d^2 - v_I{}^2 = 0 \tag{42}$$

$$t = \frac{d\cos\beta \pm \sqrt{(\cos\beta - 1)^2 d^2 + v_I{}^2}}{v_T} \tag{43}$$

Using this time we can now determine the point of intersection of the target and the interceptor. With this point, P, and the point I we can find the angle γ that the interceptor must fly at to intercept the target. By dropping lines parallel to the *x* and *y* axes from P and I we get the right triangle IAP in Fig. 13. Equations (44) and (45), in which x_T and y_T are the *x* and *y* positions of the target at the current time, v_{Tx} and v_{Ty} are the *x* and *y* velocities of the target at the current time, and x_I and y_I are the *x* and *y* positions of the interceptor at the current time, determine the angle that the interceptor must go at to intercept the target as quickly as possible.

$$\tan\gamma = \frac{y_T + v_{Ty}t - y_I}{x_T + v_{Tx}t - x_I} \tag{44}$$

$$\gamma = \arctan\frac{y_T + v_{Ty}t - y_I}{x_T + v_{Tx}t - x_I} \tag{45}$$

Our Results

Using this technique we were able to successfully intercept the target. Fig. 14 shows the paths of the target and the interceptor. It becomes very evident that the program is working correctly when the interceptor makes a sharp turn. This turn corresponds to the target's maneuver. The interceptor gets closer and closer to the target until it finally intercepts the target.

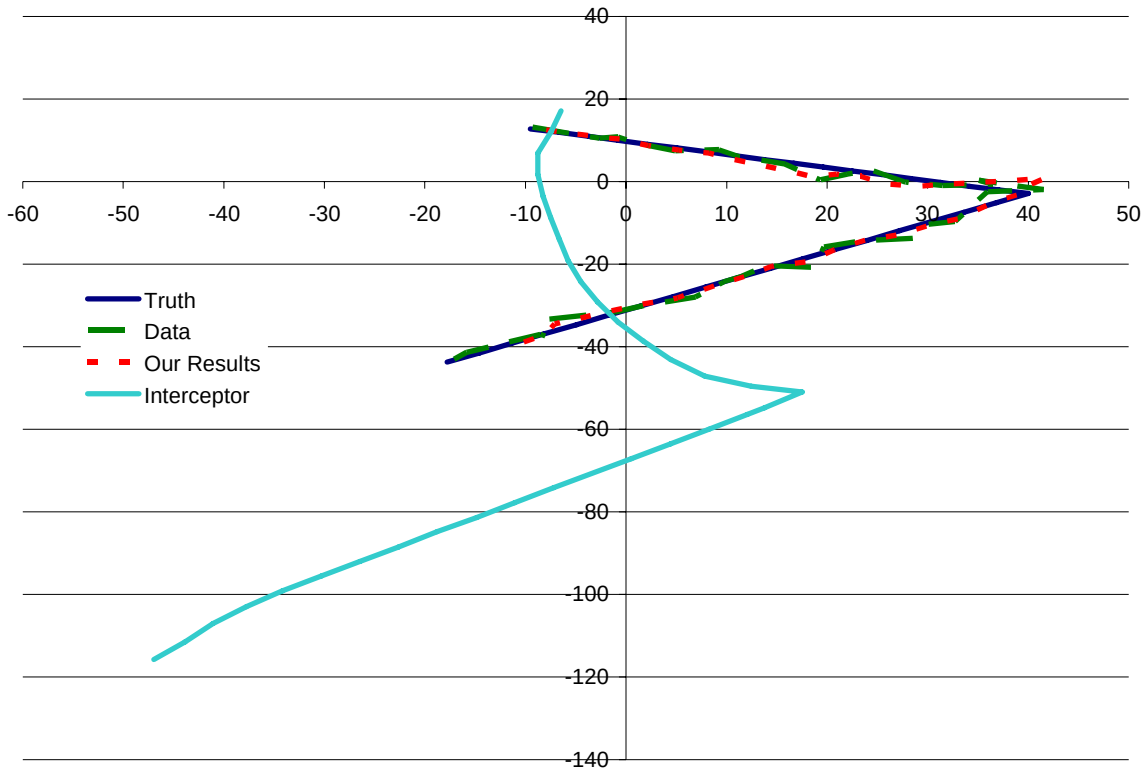

Fig. 14: Interception Path

CONCLUSIONS

Our results from the basic tracking scenarios as well as from the interception and collision prevention test cases show that our implementation of the Kalman filter is indeed effective. As intended, the filter uses measurements in real time and improves its knowledge of the target's location with each passing time step. Our filter effectively handles the state and measurement noises to provide predictions that become increasingly accurate approximations of the target's path. The filter is able to handle data coming at irregular time steps and still accurately track the target.

Perhaps the most significant result of our work is our ability to build systems that use the output data from the Kalman filter to perform important real-world tasks. Collision prevention systems like ours are extremely important in the real-world tracking systems. Air traffic control uses systems similar to ours to warn pilots of potential collisions so they can change their course. The ability to intercept targets is of huge importance for the military in its attempt to keep enemy aircraft from out skies.

Our implementation of the Kalman filter is a great model of what the Kalman filter has the power to do. This simple, efficient, and effective program is able to accurately track targets and solve complicated real-world problems.

REFERENCES

[1] Bishop G, Welch G. 2006. An Introduction to the Kalman Filter. <http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf>. Accessed 2007 Aug 8.

[2] Kalman, R. E. 1960. A New Approach to Linear Filtering and Prediction Problems. ASME Journal of Basic Engineering 1960 March.

[3] Anas SA. 2003 Jan 18. Matrix operations library .NET.

[4] Blackman SS. 1986. Multiple-Target Tracking with Radar Applications. Artech House, Inc. Norwood, MA.

[5] Atwood B. 2003. Covariance and GLAST. <http://www-glast.slac.stanford.edu/software/AnaGroup/WBA072003-Covariance.pdf>. Accessed 2007 Aug 8.