PRINCIPLES OF RADAR TARGET TRACKING

David Foregger, David Freifeld, John Kolb, Roel Mercado, Daniel Milstein,
Robert Post, Kent Twardock, Jonathan Wooldridge, Florina Yezril

Advisor: Randy Heuer
Assistant: Zachary Vogel

ABSTRACT

Radar target tracking is essential in today's world for things such as air traffic control, target tracking, collision avoidance and target interception. However, radar measurements alone are not sufficiently accurate for these purposes. This group project utilized the Kalman Filter, a mathematical algorithm that improved position estimates and provided accurate velocity vectors. The team used the Visual Basic .NET program to implement the filter. For each scenario, our results show a significant increase in accuracy of the target's position. The tracking results were used to alert aircraft of possible collisions and direct an interceptor towards a target being tracked.

INTRODUCTION

General Overview of Project

The purpose of radar is to track and predict the position and velocity of any target and to use this data for collision avoidance systems and to provide instructional data to intercept targets. However, radar measurements are inherently noisy and they do not provide an estimate of target velocity. A means of reducing the noise and providing the velocity estimate is necessary to accomplish these applications.

The project's primary purpose was to analyze and understand a specific algorithm called the Kalman Filter and apply it to hypothetical target tracking scenarios that can parallel situations in reality. The Kalman Filter is an algorithm that reduces the effect of "noise," which is a term used for inaccuracies that can be caused by a number of things, such as the inaccuracies of the radar system or the uncertainty in the motion of the target. The problem was to develop a program that applied this algorithm to a scenario and ultimately track a given target's position and velocity with greater accuracy in real time.

The Inventor of the Filter

Rudolph E. Kalman is the man who is credited with the co-creation of the filter employed in this project. He helped to integrate the use of digital computer systems to create an algorithm that stressed mathematical generality. Kalman's algorithm is one of the most widely applied techniques used in modern control theory, with uses from radar tracking aerial targets to space vehicle navigation and control [1].

Kalman was born in Budapest, Hungary on May 19, 1930. He attended the Massachusetts Institute of Technology and majored in electrical engineering. He received his Bachelor's Degree in 1953 and then his Master's Degree in 1954. Kalman proceeded on to Columbia University and acquired his Doctorate Degree in 1957. In 1958, he began his research on control theory at the Research Institute for Advanced Study (RIAS) in Baltimore [2]. Here, Kalman blended earlier fundamental work in filtering done by Wiener, Kolmogorov, Bode, Shannon, Pugachev and others with his own ideas to create a "continuous-time" version of the Kalman Filter. This version was done in collaboration with his partner R.S. Bucy [1].

Widely criticized in his field, Kalman was forced to publish his work in a mechanical rather than an electrical engineering journal. It was first used by NASA in the Apollo vehicle's navigation system and since then has been applied to a myriad of research and technological fields. One of the innovative uses is the application to modern target tracking to prevent possible collisions as well as to gain information on possible aerial threats. Its use has progressed with the advancement of the capabilities of computers and has become an essential part of modern tracking.

STATE AND MEASUREMENT MODEL

The Kalman filter was designed with the objective of reducing noise in both the state and the measurement to the greatest possible degree of accuracy. Before this objective is met, however, the models of the state and measurement must be described. Since the program's knowledge of the target is never perfectly accurate because of various sources of error, which are defined as noise, there must be some allowance for this error rather than assuming that the model is flawless.

State Model

$$x\langle k+1\rangle = \Phi x\langle k\rangle + q\langle k\rangle \tag{1}$$

The state in the context of target tracking, $x\langle k+1\rangle$, is the x- and y-positions and the x- and y-velocities of the target in question at a specific point in time. In the above model, the argument $\langle k+1\rangle$ implies that this model is the definition of a state that was predicted from the previous state $x\langle k\rangle$. This prediction is accomplished by multiplying previous state $x\langle k\rangle$ by the state transition matrix Φ. This matrix advances the state one time step, which is defined as the difference between the two times of two measurements, by having the time step entered in one or more coordinates of the matrix. It then takes into account the effect of noise by adding the matrix $q\langle k\rangle$, which represents the random state of noise instance [3].

Measurement Model

$$y\langle k\rangle = Hx\langle k\rangle + r\langle k\rangle \tag{2}$$

The measurement in the context of target tracking, $y\langle k\rangle$, is the x- and y-positions that are measured by the radar. Like the state, the measurement is susceptible to sources of error, which are represented by the measurement noise matrix $r\langle k\rangle$. The measurement transform matrix H simply transforms the state $x\langle k\rangle$ to have the same dimensions and data representation.

THE KALMAN FILTER AND ITS SIGNIFICANT MATRICES

The Kalman filter reduces noise by making a prediction of the state in the "predict" stage and then corrects its own prediction based on the actual measurement of the state in the "correct" stage. Neither the state nor the measurement is accurate because in both there is error present by the very models as defined above. Therefore, the state cannot simply be multiplied by a state transition matrix to advance it one time step, nor can the measurements simply be used as the true representation of the state. The Kalman filter will combine these two methods by predicting the state and then correcting the state based on both how accurate the filter was in the past as well as the difference between the measurement and the predicted state.

Stage I: Predict

If a very simplified view of predicting the state after one time step were adopted, the predicted state, $\hat{x}\langle k+1 \mid k\rangle$ (where $\hat{x}$ represents that the matrix is an estimate and $\langle k+1 \mid k\rangle$ indicates that the matrix is evaluated at time k+1 based on the matrix at time k) would be the current state, $\hat{x}\langle k \mid k\rangle$, multiplied by the matrix that represents the time step, Φ, as represented in Equation (3).

$$\hat{x}\langle k+1 \mid k\rangle = \Phi\hat{x}\langle k \mid k\rangle \tag{3}$$

However, there is intrinsic variability in the state. The state is not perfectly predictable because there are in fact an infinite number of factors affecting the state, all of which cannot be accounted for in any one program. Therefore, the Kalman filter includes a matrix called the state covariance matrix, $P\langle k\rangle$, which is continuously updated and accounts for the error in the state based on the previous state covariance matrix, $P\langle k \mid k\rangle$, and known statistic variability in the state, Q. In Equation (4), the current state covariance matrix must also be advanced one time step by multiplying Φ by $P\langle k \mid k\rangle$ and then by the transpose of Φ.

$$P\langle k+1 \mid k\rangle = \Phi P\langle k \mid k\rangle\Phi^T + Q \tag{4}$$

Stage II: Correct

After the predict stage, the environment has advanced one time step, and a measurement has been made for the time that was predicted. If the measurements were accurate, the measurement matrix itself would be adopted as the state matrix. However, the measurement is susceptible to error. Therefore, the Kalman filter includes a matrix called the Kalman gain, $K\langle k\rangle$, which

weights the difference between the measurement and the prediction based on the magnitude of the measurement error as represented by the *R*, where the magnitude refers to the magnitudes of all constituent coordinates of the matrix. *H* and H^T are matrices that, when multiplied by another matrix, change the dimensions of those matrices and extract specific data coordinates from those matrices.

$$K\langle k\rangle = P\langle k|k-1\rangle H^T\left[HP\langle k|k-1\rangle H^T + R\right]^{-1} \tag{5}$$

As seen in Equation (5), the value of *R* influences $K\langle k\rangle$ and thus how much the program weights the difference between the measurement and the predicted state matrix. If the measurement error is large, the Kalman gain will be smaller and will weight the measurement less. A similar case exists when the measurement error is large. These conclusions are especially significant in the next equation of the correct stage [4].

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

In Equation (6), if $K\langle k\rangle$ is sufficiently small, the prediction will be weighted more that the measurement, and vice versa.

$$P\langle k|k\rangle = \left[I - K\langle k\rangle H\right]P\langle k|k-1\rangle \tag{7}$$

The state covariance is then corrected based on the value of the Kalman gain, as given in Equation (7). If Kalman gain is small, the resulting change in the state covariance matrix will be small. Since *I* is the identity matrix, subtracting the product of $K\langle k\rangle$ and *H* from *I* will be the weighting factor to form a new state covariance for the next time step [3].

PROGRAMMING THE FILTER

The filter was programmed using Visual Basic .Net to allow for Object Oriented Programming. Using the general concept of objects in Visual Basic allows the programmer to apply certain specific functions to multiple objects of an entire class, thus generalizing the program for more practical use. It would be possible to create new matrices for each and every new target that must be tracked, but the code in the main module would be incredibly dense and unorganized. In the case of tracking multiple targets, creating a broad class and then instantiating an object of that class for each target is much more practical than the technique described above. Creating a class enables the programmer to apply certain functions that have been described in that class to multiple objects. A function that has been created in the class can be called upon in the main module, applied to several objects, and assign several values that will be used in the main module. This technique of programming is more efficient and organized than establishing new matrices within the module for each target.

Structure:

The program is structured in a mostly linear fashion. It begins by creating an object of the class Target and by initializing the variables it needs to keep track of the interceptor, targets, and time. Next it enters a loop which is used to go through all the given data and apply the Kalman filter to it. Once there are no more data points, the loop is exited and the data is printed to the front page.

The program relies heavily on the MatLib class [5]. MatLib makes it very simple to add, subtract, multiply, transpose, and inverse matrices. This cleans up the code substantially and avoids including the nuts and bolts of matrix operations within the body of the code.

The Loop:

The loop performs two functions: reading the data in and utilizing it. First it calls the function getData() which reads the next line of the file, splits it into a 1-dimensional array with time and the polar coordinates, and then returns a 1-dimensional array with time, x-y coordinates, and polar coordinates. If there is no line to read then the loop is exited. Next the data read in is pass to a sub utilizeData(). This sub begins by passing the data read in to the addPoint() function of the target. When addPoint() is called for the first time, an array is returned with a null data set because there is not enough data to predict or correct any values yet. The second time addPoint() is called, it initializes the arrays being stored in Target (see below) and again returns a null data set. In all subsequent calls, addPoint() edits the R, Φ, and Y matrices, calls Predict(), calls changeDir() to check for a change in direction, and calls Correct(). Lastly, an array with the time, x value, x velocity, y value, and y velocity is returned to utilizeData(). This data, unless it is a null set, is added to a string for future printing, added to an array for drawing the path, and passed to a function SetPlaneVelocity(). SetPlaneVelocity() uses an algorithm to determine the optimal direction of the interceptor and calculates the x and y velocities it should fly at. Next, a module is used to determine if the interceptor has reached the target and if so, stops looping and appends to the results that the target has been intercepted. Finally, the time and location of the target are updated.

The Target Class:

The Target class keeps track of all the matrices that are used in the Kalman Filter. This is vital in the specific case of multiple target tracking. By using the Target class, all the matrices are stored individually for each target and are kept track of outside of the main code. The diagram below represents how the class can create multiple objects with the matrices self-contained. All the interaction between the main class and the targets can come through AddPoint(), as diagrammed in Figure 1. In the diagram there are only two targets, but theoretically any number of objects can be created. This is a good demonstration of the usefulness of Object Oriented Programming.

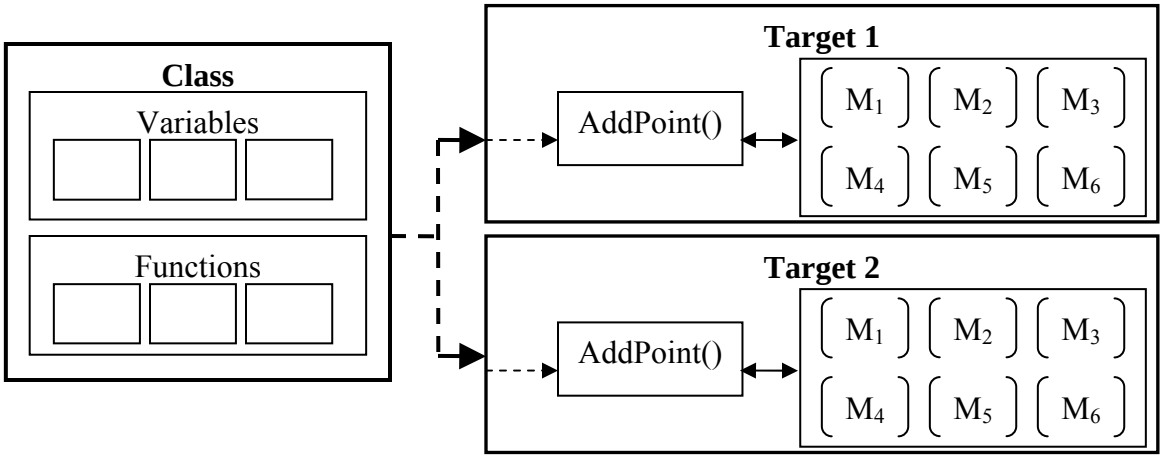


Fig. 1: Diagram of Object Oriented Programming with Target Class

CASE COMPENDIUM

- Polar Coordinate Tracking
- Tracking with Multiple Radars
- Tracking Targets with Changing Velocity and Direction
- Intercepting a Target
- Tracking Multiple Targets
- Aircraft Collision Avoidance

These six scenarios are the ones that demonstrate the capabilities of the Kalman Filter and the Visual Basic program. The cases become more difficult as they progress and they encompass the gradual increase in the complicating factors of the targets. As each scenario was received, the group had to adjust their programs to meet the conditions. Each addition and change to the program also made it more generic to allow it to track the new target with little change to the basic coding. The simplest case began with a data set simply given in Cartesian coordinates and they moved on to Polar Coordinates – which is the system used in actual radars. Subsequent scenarios involved targets that changed direction, the use of more than one radar, and finally the simultaneous tracking of multiple objects.

The basics of the project began with the tracking of 2-dimensional objects on a Cartesian plane that were measured in x and y directions with respect to the radar at the origin. Later scenarios involved multiple targets and radars, objects that change direction, and finally a switch to the calculation of incoming data on a Polar Coordinate plane. This polar system involved tracking the target a specific number of degrees from “true North” and a certain range with respect to the origin.

After each scenario was completed, the team proceeded to do a comparative analysis of our predictions and estimates with those of the “true” flight path and velocities of the target that was tracked. The purpose of this analysis was to determine the amount of error that the filter had produced and to see how accurate the calculated data was. The final goal of the project was to

create a generic program that could function for any type of situation and number of targets. Overall, the Kalman Filter's job was to analyze the incoming measurements from the original radar and correct them to provide the users with closer results.

SCENARIOS AND CASES

Polar Coordinate Tracking

The radar in this case study is tracking a commercial plane solely for the purpose of predicting its flight path. The basic example of a Kalman Filter operates using a Cartesian system. However, most radars do not provide measurements in Cartesian coordinates. The radar here measured the distance from the plane, r. It also determined the bearing of the plane from North. The challenge here was in transforming measurements and measurement uncertainties into useable data.

Transforming Polar to Cartesian Data

Transforming Coordinates:

The basic component in formatting the given data to a more usable state was in converting data from polar to rectangular coordinates. Using trigonometric operations with r and the given angle, the data became a pair of coordinates in an (X, Y) format. Equations (8) and (9) define the transformations. These updated coordinates are easily inserted into the previously defined matrices to allow for the use of the original Cartesian system while doing calculations. The bearing was given in degrees from North; the program accounted for this by converting θ. By just editing this part, the original X-Y coordinate program was conserved. This proved the efficiency and adaptability of the initial program.

$(r, \theta) = (X, Y)$, where

$$X = r \cdot \cos\theta \qquad (8)$$

$$Y = r \cdot \sin\theta \qquad (9)$$

For instance, when r = 30 and θ=30º from North,
First convert θ=30º, to 60º from the x-axis then

$$X = 30 \cdot \cos 60 = 15$$

$$Y = 30 \cdot \sin 60 = 26$$

(30, 60º) becomes (15, 26)

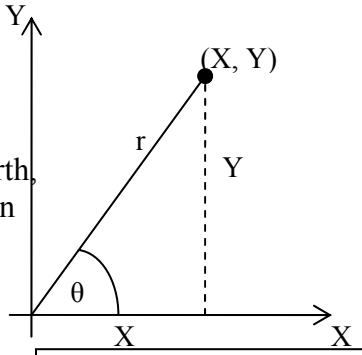
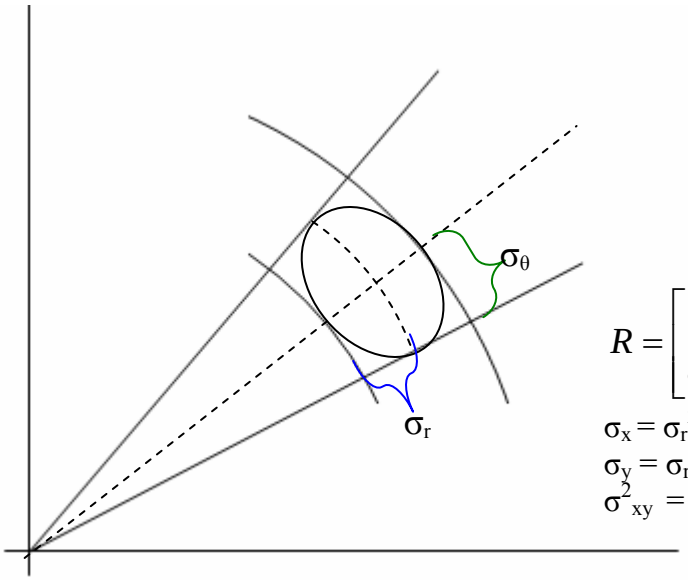


Fig. 2: Visual Representation of Polar Coordinates

Transforming Error:

The second aspect of this conversion involves the measurement error. In a rectangular system, a circle or ellipse will surround the predicted point. The point of estimate will be in the center of the ellipse, with major and minor axes parallel to the X-Y coordinate system axes. Most probably, the true location of the plane will be within this area.

However, this situation is complicated by the polar measurements. The given situation declares a range error of one mile and a bearing error of three degrees. Instead of a circle or an ellipse with major and minor axes are parallel to coordinate system axes, the error produced will be in the shape of an approximated ellipse oriented along a different axis as in Figure 3. An updated R matrix was required. The error of bearing σ_θ and the error of distance σ_r were converted into σ_x and σ_y terms to relate the variation. The covariance terms σ^2_{xy} were both zero in situations with rectangular coordinate radar data because the x and y were independent. In the polar case the σ^2_{xy} are nonzero to account for the distortions of the ellipse. The R matrix is also displayed in Fig. 3.



$$R = \begin{bmatrix} \sigma_x^{\,2} & \sigma^2_{xy} \\ \sigma^2_{xy} & \sigma_y^{\,2} \end{bmatrix}$$

$\sigma_x = \sigma_r^2 \cos^2\theta + R^2 \sin^2\theta \, \sigma_\theta^2$
$\sigma_y = \sigma_r^2 \sin^2\theta + R^2 \cos^2\theta \, \sigma_\theta^2$
$\sigma^2_{xy} = ½ \sin(2\theta) [\sigma_r^2 - R^2\sigma_\theta^2]$

Fig. 3: Polar Measurement Error Range and Matrix

Results

In Target Position Analysis Figure 4, the solid gray line of raw data showed noise inherent in the measurements. The estimates are shown as a solid black line; they started at left by changing as the raw data fluctuated. Toward the right, the estimate approached the dashed gray truth line. The estimate remaining fairly constant even as the radar data varies, especially at the end of the program because of the effective Kalman filter.

The case position residual graph (Figure 5) demonstrates these results. Each position residual represents the discrepancy between the estimated and measured locations of a target for a specific time. Following the first point, the point predicted was always within one mile of where the plane was at the next time step. For the time from 0 to 0.125 hours clearly demonstrates the Kalman filter at work. Despite changes from point to point, the trend is for the discrepancy to decrease. Even around 0.175 hours when the noise projected that the plane was far off course, the predicted estimate was within ¾ of a mile; the measurements were almost three times as far.

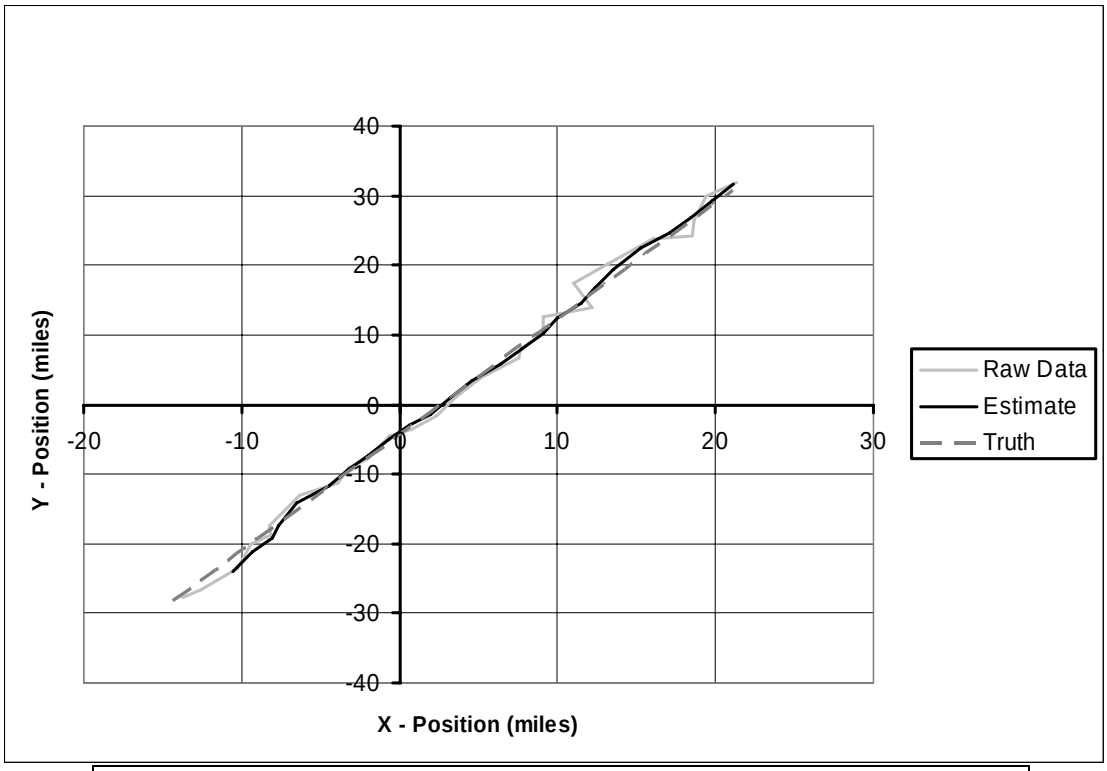


Fig. 4: Target Position Analysis – Filter with Polar Coordinates

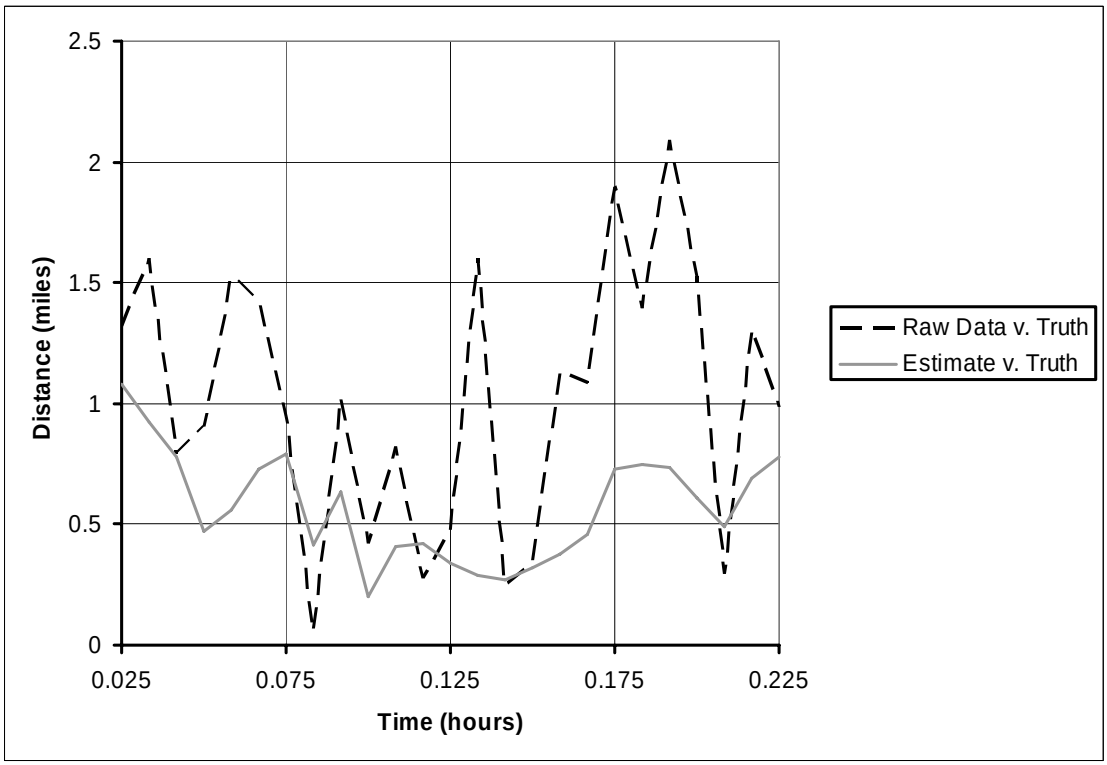


Fig. 5: Position Residual - Filter with Polar Coordinates

Tracking with Multiple Radars

Having conquered the issue of data present in polar coordinates, the fourth case further reflected real-life situations of target tracking. Here, the location of an Unidentified Flying Object (UFO) was being observed by two different radars. In prior situations, the radar location was at the origin of the coordinate system being used to measure the target. In this situation however two radars cannot be co-located; each is at a location other than the origin. Unlike prior programs, the data readings no longer were taken on even intervals of time. The program was altered to reflect these aspects of the new case.

Accounting for Separate Radars:

Data being received by different radars must be sorted prior to being processed. Using the Visual Basic program made this task easier. A simple IF-THEN statement allowed the program to identify whether the coordinates came from one radar or the other.

Accounting for Separate Radar Locations:

Once the distinction was made between radars, the program had to account for the discrepancy in distances. For each of the radars the data was being measured as though it was at the origin of its own system. Knowing the different locations, the frame of reference of each could be determined. This process was performed by subtracting the radar coordinates from the projected UFO coordinates to achieve points in the same frame of reference. Now the data could be effectively used in the Kalman filter.

Accounting for Variable Time Steps:

Another aspect of this problem was the variable time step. In earlier problems the change in time was constant between measurements of the target location. The program now had to be changed from using a constant value to using the variable. At each step the new change in time was calculated by subtracting the prior time from the 'current' and saving that amount to an updating variable. This variable was employed in the program instead of the constant.

Results:

The Target Position Analysis graph in Figure 6 shows the progress of the Kalman filter being used with two radars. The UFO started in the lower right corner of the graph and proceeded north-west. In the middle of the course, the data deviates from the truth. As the end of the run approached, the estimate gained accuracy.

The Position Residual in Figure 7 is an exemplary case of the Kalman filter at work. The inaccurate reading at $t = 0.105$ hours influenced the estimate, but by the time that occurred the prior readings balanced this. The reading was almost seven miles from the truth, but our estimate was only about three. From that time, the estimates continue to increase in accuracy. While the radar readings at the end are never within one mile of the truth, the predictions are more accurate and appear to reach zero.

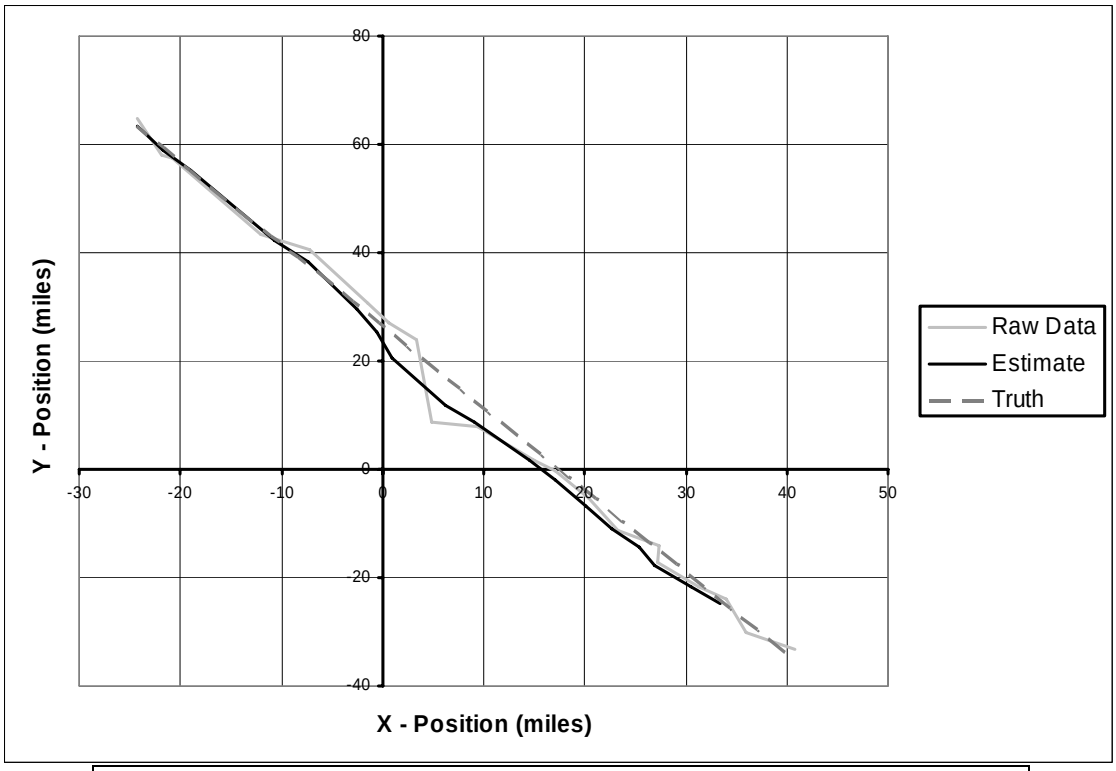


Fig. 6: Target Position Analysis – Tracking with Multiple Radars

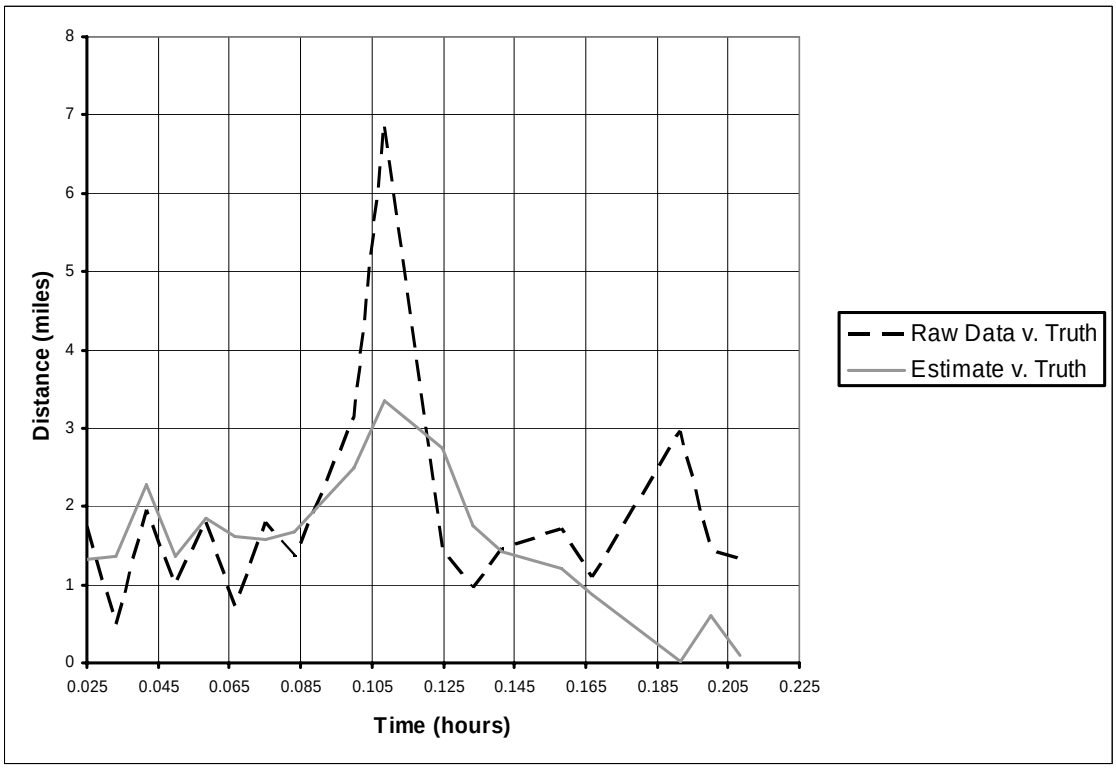


Fig. 7: Position Residual – Tracking with Multiple Radars

Tracking Targets with Changing Velocity and Direction

Problem:

In a real life scenario, not all targets will be traveling in a straight line. Due to noise, it is not always simple determining when a target has changed its path; statistically a value outside of the predicted path does not necessarily mean a change in direction. This adds complexity because every change in direction lowers the accuracy of the filter and false reports of a changed direction weakens the effectiveness of the filter.

Solution:

In order to deal with changing velocity and direction, a check must be performed every time step after the predict stage and before correcting. The error, in number of standard deviations (n), can be described with Equation (10).

$$n = \sqrt{Z^{-1} C^{T} Z} \tag{10}$$

Where $Z = \begin{bmatrix} x_{predicted} - x_{radar} \\ y_{predicted} - y_{radar} \end{bmatrix}$ $C = \begin{bmatrix} x_{variance} & covariance_{xy} \\ covariance_{xy} & y_{variance} \end{bmatrix}$

When the error n is greater than five standard deviations for three consecutive time steps, then it is assumed that the target has most likely changed direction. Statistically speaking, there is less than a 1% chance a point will be outside five standard deviations. Once could be a result of an extreme error in the radar reading, but three readings are an indication of course change. All the matrices are reinitialized to their values as if the target started flying one time step earlier.

Results:

The results for this calculation were very successful. Figure 8 shows how the tracker predicts past where the target changes direction (A) before determining the direction has changed and restarting the filter (B). It can be seen in Figure 9 that the error in estimate compared to truth spikes up at points of acceleration (C, D) before the filter grows more accurate again [3].

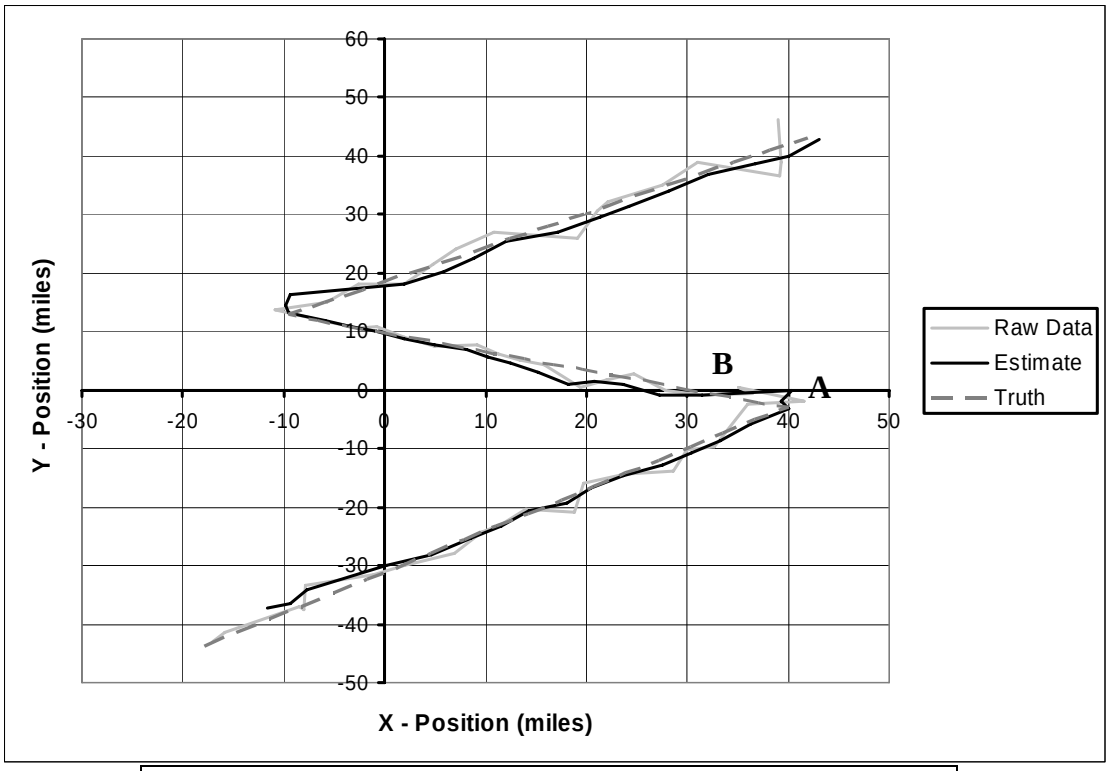


Fig. 8: Target Position Analysis - Maneuver Detection

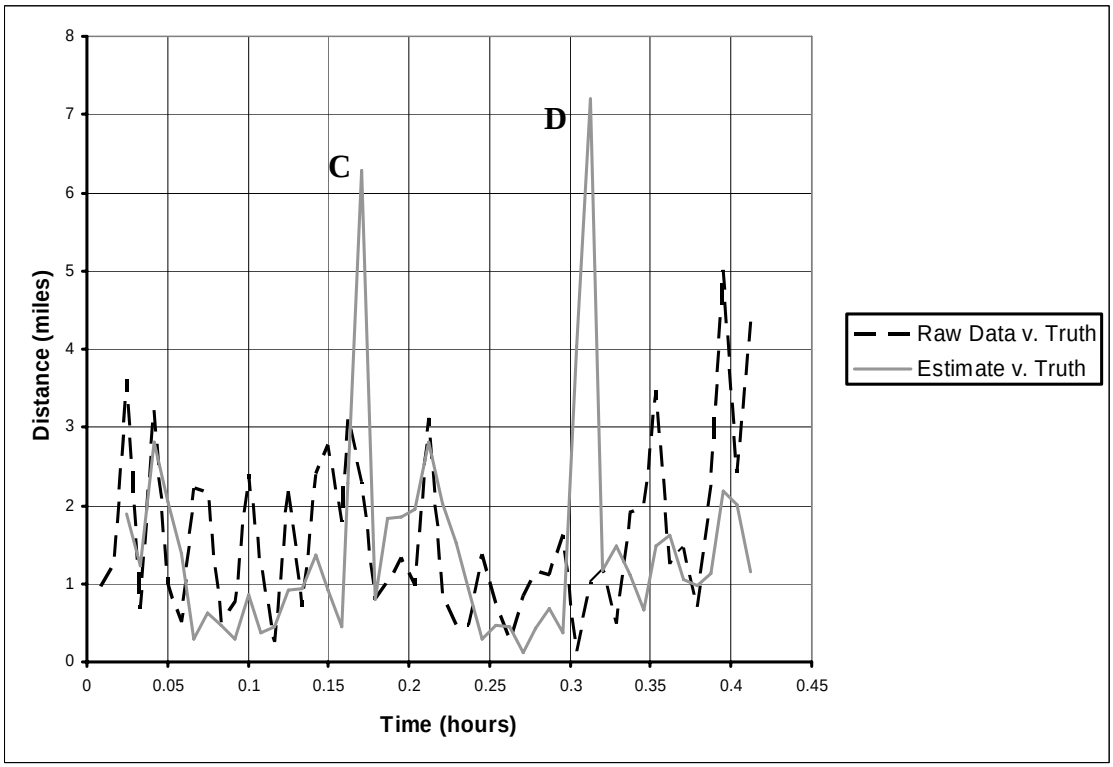


Fig. 9: Position Residual - Maneuver Detection

Intercepting a Target

Problem:

Often, a target found will need to be intercepted, such as an incoming plane by a fighter jet. The question that is presented is where the intercepting device should fly to intercept the target in the shortest time possible.

Solution:

The solution to this is to create a triangle (Figure 10) between the intercepting device (I), the current position of the target ($T_{current}$), and the unknown point ($P_{intercept}$) as well as another triangle between the intercepting device, the current position, and the next position (T_{next}). Figure 10 shows this.

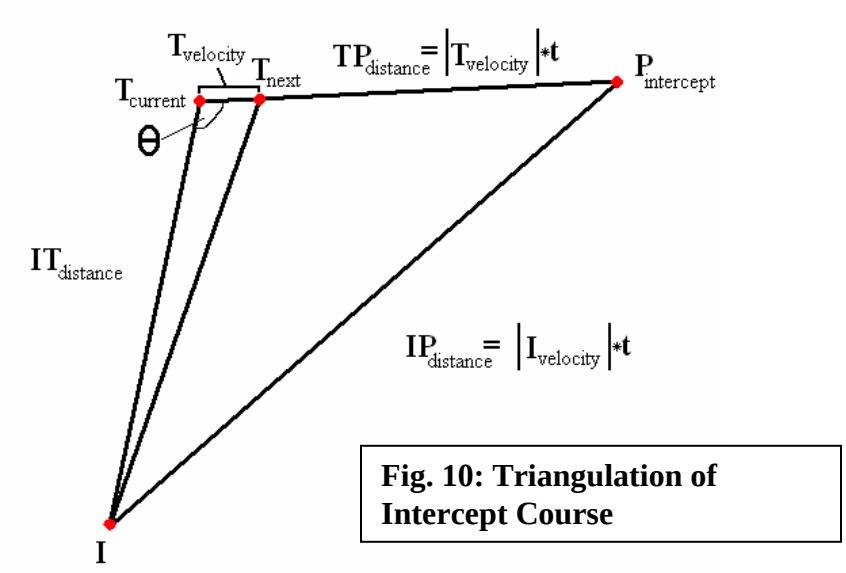


Fig. 10: Triangulation of Intercept Course

The distance $IT_{distance}$ between the interceptor and the target can be determined by simply applying the distance formula. The angle $I_{current}T_{current}P_{intercept}$ (called θ) can be found using the distance formula combined with the law of cosines on $I_{current}$, $T_{current}$, and the next position of the target (T_{next}).

Knowing the angle θ, it becomes possible to find the time (t) until intercepting the target. The distance $TP_{distance}$ between the target and $P_{intercept}$ can be defined as $|T_{velocity}| * t$ where $T_{velocity}$ is the velocity of the target and t is the time predicted until they intercept. Additionally, the distance $IP_{distance}$ between the interceptor and $P_{intercept}$ is $|I_{velocity}| * t$ where $I_{velocity}$ is the velocity of the interceptor. By applying the law of cosines using θ, TPdistance, $IP_{distance}$, and $IT_{distance}$ it is possible to create the quadratic equation shown in Equation (11) and solve for t.

$$0 = t^2 * \sqrt{(I_{velocity})^2 - T(_{velocity})^2} + t * 2IT_{distance} T_{velocity} \cos(\theta) - IT_{distance}^2 \quad (11)$$

As many as two values are obtained when Equation (11) is solved for t. Each of the two can be either negative or positive. In the event that both points are negative or the roots of t don't exist, the two objects will never intersect. If only one of the values produced is positive, it chosen for use in the remaining part of the step. If both are positive, two possible paths exist for intersection; the smallest value is chosen as the shortest time the intercepting device must travel to meet its target. This can then be plugged back into $TP_{distance}$ and $IP_{distance}$ to determine the optimal point of intersection and consequently the direction which the intercepting device should travel.

This method is performed at every stage in order to update the anticipated point of interception as the filter gets more accurate and the estimated velocity of the target changes.

Results:

The results of this formula can be seen Figure 11. This example uses an intercepting device with a velocity of 630 MPH. During the first stage of the simulation, the interceptor flies northeast towards the target's presumed heading. When the target changes direction at A_1 the interceptor realigns its heading towards the new heading of the target A_2. The curve of the path of the intercepting device from A_2 to the interception point is evidence of the readjustment of the interceptor's path as the filter gets more accurate. The program ends as the intercepting device is within one mile of the target. At that point the target would be in sight of the plane and radar tracking has fulfilled its purpose.

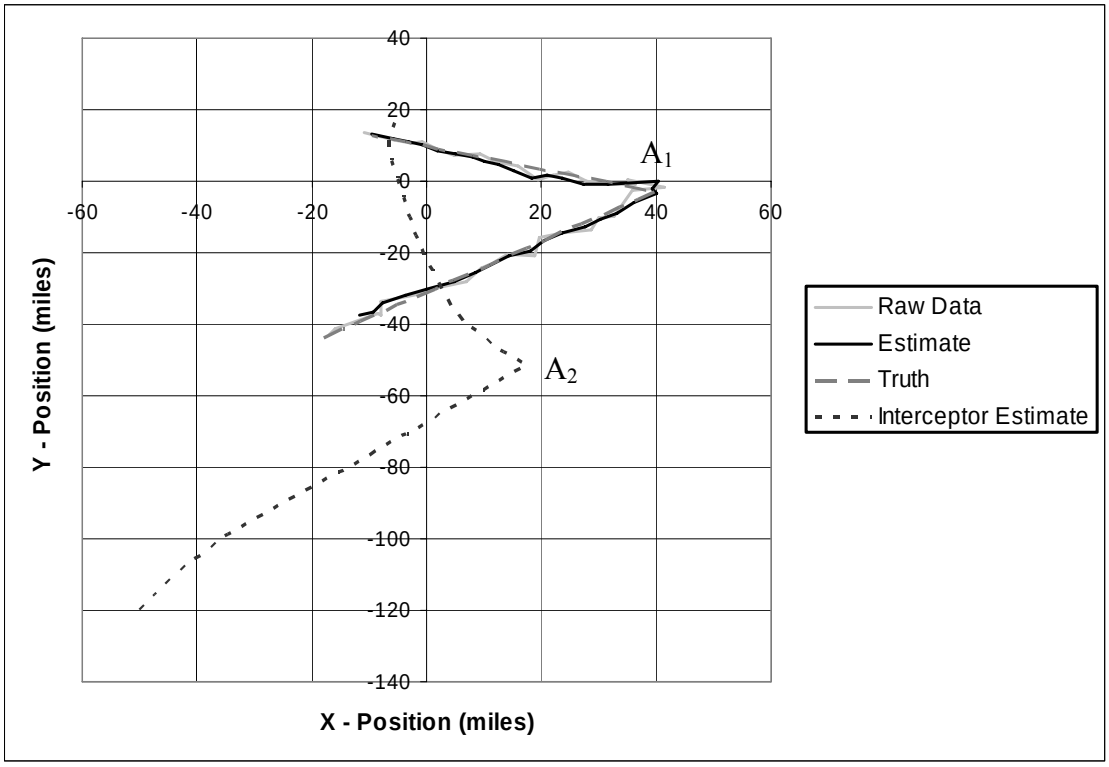


Fig. 11: Target Position Analysis – Intercepting a Target

Tracking Multiple Targets

Another practical application of the Kalman filter is to track multiple targets. Developing a program that will only track a single target ignores the fact that a radar mechanism in applied use will actually be tracking several airplanes, and it disregards the possibility that two targets will collide if they maintain the same course. It is therefore necessary to develop an algorithm that will track the positions of several targets, predict intersections of the paths of two targets, and alert those specific targets on course for a relatively likely collision.

Thus far the programs described above have not taken into account the chance of two targets colliding, which is one of the foremost reasons for using target tracking programs. At high velocities, these targets will not be able to adjust their course in time to avoid a collision if they rely on individual vision alone. It would be much safer to predict the course of multiple targets and generate an alarm if, at the predicted of intersection of two targets' courses, the two targets were within a mile of each other, which is the standard deviation of range error.

Preparing the Program to Alert:

It will not be necessary to calculate a point of intersection if the two targets are not reasonably close to each other. The distance between the targets can be easily calculated using the Distance Formula, and if the distance were less than an arbitrary amount then the program would calculate a point of intersection. This cutoff distance, which was determined to be twelve miles, takes into account that at high speeds the pilots of the targets in question would require approximately ninety seconds to appropriately alter their course once alerted.

Calculating the Point of Intersection (POI):

Once the distance between the two targets is determined to be less than twelve miles, a POI must be calculated in order to determine if the targets are in danger of colliding. The POI can be calculated can by treating each target as point on a line with the quotient of the y-velocity and x-velocity as the line's slope. The POI can be calculated by solving the simultaneous linear equations characterized by the aforementioned slopes and the positions of the targets, as seen in the procedure below. Note that Equations (12, 13, 14) utilize the expressions (x_{POI}, y_{POI}) and ($x_{coord,\mathrm{n}}$, $y_{coord,\mathrm{n}}$), where (x_{POI}, y_{POI}) are the coordinates of the POI and ($x_{coord,\mathrm{n}}$, $y_{coord,\mathrm{n}}$) are the coordinates of plane n.

$$\begin{cases} y_{POI} = \dfrac{y_{vel,1}}{x_{vel,1}}\left(x_{POI} - x_{coord,1}\right) + y_{coord,1} \\ y_{POI} = \dfrac{y_{vel,2}}{x_{vel,2}}\left(x_{POI} - x_{coord,2}\right) + y_{coord,2} \end{cases} \tag{12}$$

$$\frac{y_{vel,1}}{x_{vel,1}}\left(x_{POI} - x_{coord,1}\right) + y_{coord,1} = \frac{y_{vel,2}}{x_{vel,2}}\left(x_{POI} - x_{coord,2}\right) + y_{coord,2} \tag{13}$$

$$x_{POI} = \frac{\frac{y_{vel,1}}{x_{vel,1}} x_{coord,1} - \frac{y_{vel,2}}{x_{vel,2}} x_{coord,2} - y_{coord,1} + y_{coord,2}}{\frac{y_{vel,1}}{x_{vel,1}} - \frac{y_{vel,2}}{x_{vel,2}}} \tag{14}$$

Determining if an Alert Is Necessary:

Even if the targets are twelve miles apart, they are in no danger of colliding if at no point in time will the two targets are predicted to be within one mile of each other. It is then necessary to determine if the respective targets truly must be alerted. The time interval to when one target will reach the point of intersection can be calculated by using one target's twelve-mile and POI x-coordinates and the x-velocity of that one target. These are applied to Equation (15) to solve for the desired value.

$$t = \frac{x_{POI} - x_{coord,1}}{x_{vel,1}} \tag{15}$$

Once the time step is known, the position of the other target can be calculated using its respective x- and y-velocities. Note that Equations (16, 17) utilize the following expression ($x_{final,2}$, $y_{final,2}$) where ($x_{final,2}$, $y_{final,2}$) are the coordinates of the second target when the first target is at the POI.

$$x_{final,2} = x_{vel,2} t \tag{16}$$

$$y_{final,2} = y_{vel,2} t \tag{17}$$

The Distance Formula can then be used in order to determine whether the targets will indeed be within a mile of one another. This procedure must be repeated using the other target to calculate an alternate time interval.

Data Analysis and Position Residual:

Since the program must process a significant amount of data to reasonably calibrate its matrices, the first several estimates are not accurate (Fig. 12, 13). As time progresses, however, the error of the Kalman-filtered estimate with respect to the truth approaches zero and eventually becomes more accurate than the raw data. An alert is issued well before the predicted time of intersection, which would in practical use give the pilots of the targets enough time to alter their respective courses.

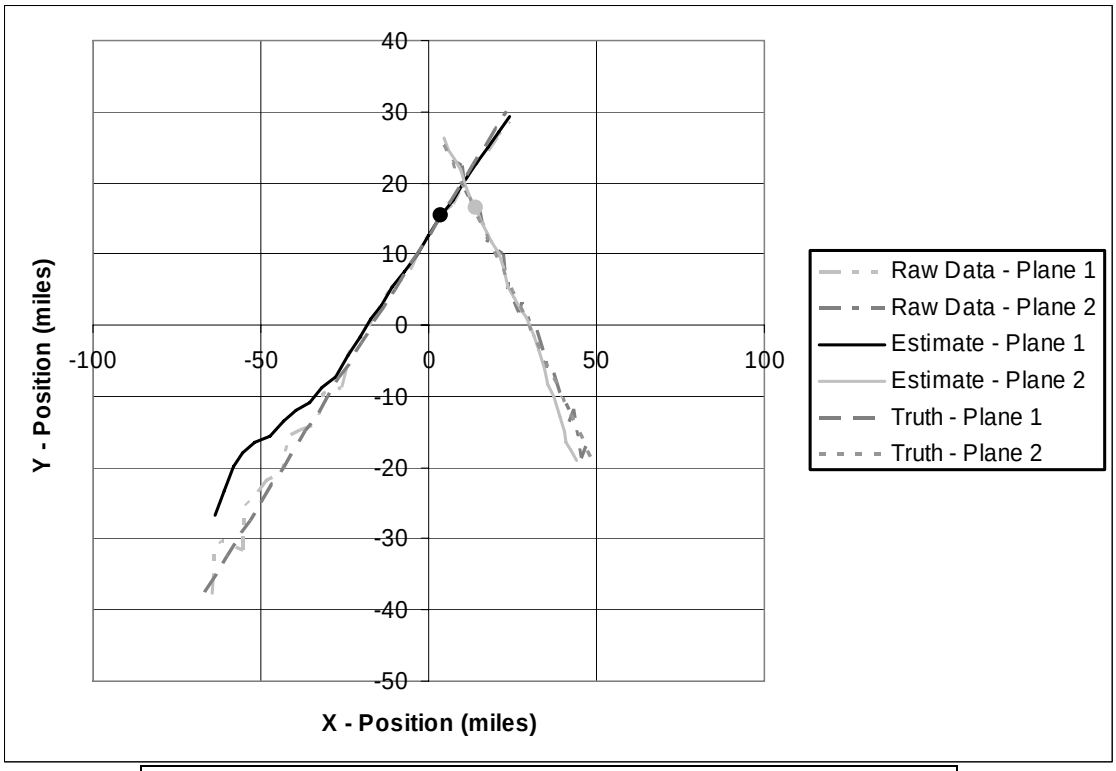


Fig. 12: Target Position Analysis - Collision Avoidance

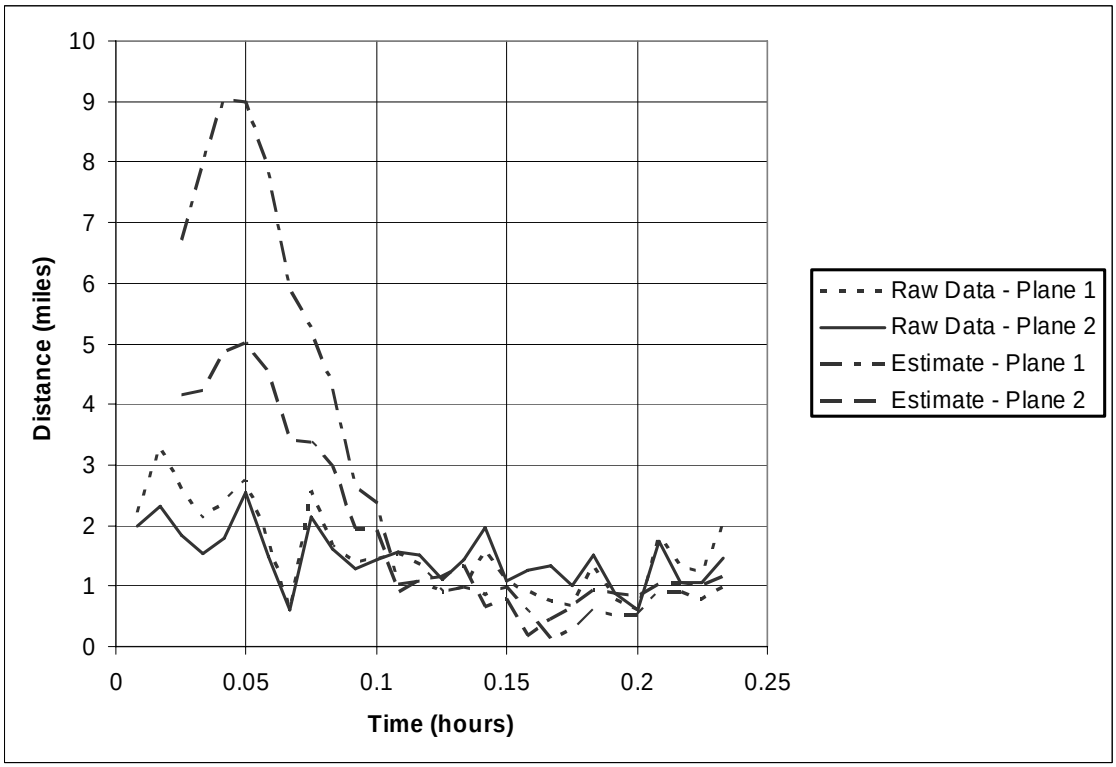


Fig. 13: Position Residual - Collision Avoidance

CONCLUSION

In our project, we created a Visual Basic .NET computer program implementing the Kalman Filter to track and predict the motion of radar targets. The cases we processed progressed from tracking simple, linear motion in a Cartesian plane to more complex scenarios. Our Cartesian-measurement model evolved to track a target whose position was measured in range and bearing. The next scenario incorporated multiple targets. Using our multiple-target implementation, we progressed to the early detection and avoidance of possible aircraft collisions. Our next challenge was to detect and respond to target maneuvers so to prosecute a non-linear motion path. Our final scenario incorporated our maneuver detection algorithm and involved tracking and intercepting a maneuvering target.

In all cases, the Kalman Filter demonstrated noticeable improvements over raw measurement data, improved accuracy over time and was easily adaptable to more complex situations. Ultimately, the results show that the filter provides a simple and effective technique to calculate relative velocities, positions, and projected flight paths of radar targets.

REFERENCES

[1] [IEEE] Institute of Electrical and Electronics Engineers. 2003 Jan 23. Rudolf E. Kalman, 1930-. IEEE History Center. <http://www.ieee.org/web/aboutus/history_center/biography/kalman.html> Accessed 2006 Aug 3.

[2] Department of Computer Science at University of North Carolina. 2001 Jan 31. Rudolph Emil Kalman. <http://www.cs.unc.edu/~welch/kalman/kalmanBiblio.html> Accessed 2006 Aug 3.

[3] Blackman, Samuel S. 1986. Multiple-Target Tracking with Radar Applications. Artech House, Inc. Norwood, MA.

[4] Bishop G, Welch G. 2006. An Introduction to the Kalman Filter. <http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf>. Accessed 2006 Aug 3.

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