

# Improving Aviation Safety using Low-Cost Marine Radars with Extended Kalman Filters in Aircraft Sense and Avoid Systems

Rohan Deshpande

under the direction of  
Jae-Woo Choi, Franklin Wu, and Michael Sardonini  
Aurora Flight Sciences

Research Science Institute  
August 3, 2017, revised September 4, 2017

## **Abstract**

In recent years, congestion in the global airspace has been increasing at an exponential rate, increasing the probabilities of an air collision. Currently, transponder based collision avoidance systems fail when other aircraft do not have compatible transponders systems installed. To reduce reliance on a transponder based collision avoidance system, a low cost radar-based collision avoidance system for aircraft was developed. The tracking system was developed using memory efficient cluster identification algorithms to identify nearby aircraft, state estimation equations to develop track associations to differentiate between multiple nearby moving aircraft, and Extended Kalman Filters for aircraft position and velocity estimation refinements. During the testing phase, aircrafts were accurately tracked in the local airspace, demonstrating the low-cost radar system's feasibility. Modern Collision Avoidance Systems for aircraft cost about \$20 million, while the developed Sense and Avoid radar system costs only \$20,000. Thus, we demonstrate that the cost of aircraft collision avoidance systems can potentially be reduced by a factor of 1000.

## **Summary**

The Air Traffic Control (ATC) is tasked with guiding an array of airplanes, each traveling to different destinations, through a controlled airspace. The ATC strives to control the aircraft in an efficient manner to avoid delays, while ensuring passenger safety. Due to increasing air traffic congestion, the probability of collisions has significantly increased. Thus, it is essential that all aircraft, whether they are large commercial aircraft or small private aircraft, are equipped with appropriate Collision Avoidance Systems. In this paper, a low cost radar-based collision avoidance system is developed and tested with the hope of constructing a collision avoidance system that will eventually be installed in aircraft in the near-future.

# 1 Introduction

In recent years, congestion in the global airspace has been increasing at an exponential rate, mirroring an increase in demand for airline flights (Fig. 1). During daily peak operations, as many as 8,000 aircraft may be operating simultaneously in U.S. airspace [1]. To reduce the chaos in the air, the Air Traffic Control (ATC) is tasked with guiding airplanes safely through the skies. With this many aircraft in flight at once, congestion and resulting flight delays are inevitable. Furthermore, many companies hope to use networks of small unmanned drones for numerous applications in the near future. This will further increase the congestion of in the airspace, specifically in the lower altitudes where aircraft are taking-off and landing.

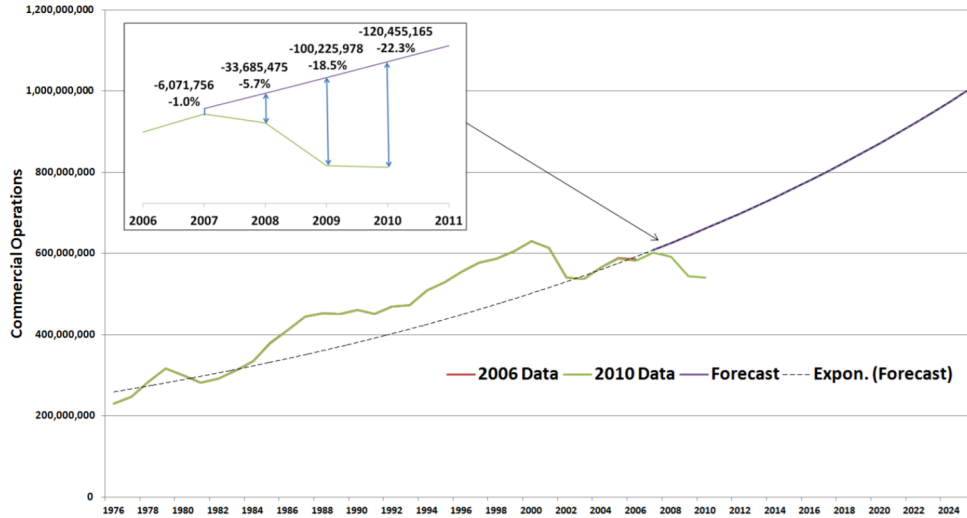


Figure 1: Airline Commercial Operations Forecast [2]

One of the current solutions to mediate the air traffic congestion is to allow multiple planes to fly over each other when their trajectories intersect. Depending on the altitude difference between the two (or more) aircrafts when they cross paths, this behavior can be quite risky. As the altitude difference decreases, there is less margin for error, which could be caused by pilot error or inaccurate altitude measurements. In the event that there are

multiple aircraft trajectories set to intersect due to improper altitude control, an on board collision avoidance system is essential. However, the high cost of collision avoidance systems is often motivation for not installing these essential Sense and Avoid Systems on smaller private aircraft or drones. This causes these aircrafts or drones to act as “ghosts” – undetectable to many collision avoidance systems because the systems rely on both aircraft having the correct transponders installed. There are many benefits to using radar systems to detect other flying objects, instead of transponders. One such benefit is that radar can easily detect other projectiles which appear in the airplane’s airspace. This property is extremely useful for larger aircraft, as the radar system will allow them to detect smaller ghost aircraft. The radar system can also benefit the ghost aircrafts: the cost reduction would motivate smaller aircraft or drones to install the low cost radar system if they do not already have modern sense and avoid systems installed. Thus, there is a need for a low cost, radar-based collision avoidance system for aircraft.

The inspiration for the research on cost reduction of sense and avoid systems on aircraft emerges from the automobile industry. Over the last decade, the concept of autonomous cars has gone from a prototype phase to a commercially viable product; companies, including Tesla Inc. and Google Inc., have demonstrated functional prototypes of cars autonomously driving on highways. In its current form, Tesla has successfully packaged its autonomous car software onto its current cars, labeling the software as a “Driver Assistance System.” Similar to Tesla’s driver assistance system, the aviation industry has also built systems to assist pilots during the flight, once the aircraft has reached its cruising altitude.

The giant leap Google made, from demonstrating an expensive but functional prototype to actually releasing its driver assistance software in their current Tesla car models, was accomplished through numerous cost cutting strategies; in other words, the cost of the technology had to be reduced for the product to be considered as commercially viable. For example, the Google Driverless Car used a Light Detection and Ranging (LIDAR) system

to generate a 3D map of the car's surroundings. The LIDAR system utilized on their initial prototype cost \$75,000 [3]. One of Google's strategies was to replace the expensive and bulky LIDAR system with less expensive optical imaging technologies.

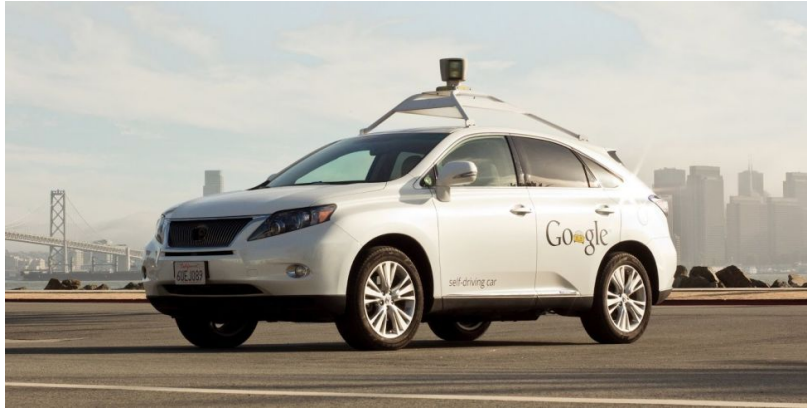


Figure 2: Google Driverless Car – the black cylindrical object on top of the car is the LIDAR system which generates a 3-D map of the car's surroundings [3]

Similarly, Google's effective cost reduction strategy can also be applied to the aviation industry. Rather than using numerous astronomically expensive sensor devices, the cost can be reduced by applying modern marine radar systems to aircraft. To combat the reduction in the accuracy of the input data (due to the cheaper sensors), the input data can be filtered by exploiting the statistics of the data set in real-time.

Modern commercial aircraft use multiple sensors for Sense and Avoid systems as they are critical in mid-air aircraft collision avoidance. Currently large commercial aircraft have a Traffic Alert & Collision Avoidance System (TCAS), which includes transponders that constantly sends "pings." The pings, which operate under the radio wave frequency spectrum, allows aircraft in the vicinity to be notified of incoming air traffic and its approximate locations. Using the "pings" and approximate locations, both aircraft are able to make corrective vertical/horizontal maneuver assigned to both planes to avoid a collision. New regulations mandate that large aircraft must also have an Automatic Dependent Surveillance

Broadcast system (ADS-B) installed by 2020. This newer technology involves broadcasting the aircrafts position, speed, heading, and other critical parameters [4]. When combined with the modern weather radar, the cost of these three critical technologies installed on most large aircraft is approximately \$20 million [5]. To put this into perspective, the cost of these three technologies accounts for 10% of the total cost of Boeing 787-9 Dreamliner airplane. The significant cost of the Sense and Avoid Systems serve as motivation for a novel cost reduction strategy that does not compromise aviation safety.

The application of Sense and Avoid object detection systems is not limited to commercial passenger and military aircraft – they are also commonly used systems on modern Unmanned Aerial Vehicles (UAVs)[6]. With companies such as Amazon, Facebook, and Google proposing the use of a large network of UAVs, it is essential that these UAVs are equipped with the collision avoidance systems as they have to share airspace with airplanes. New UAV regulations have proposed a radar view of 220° horizontal by 40° vertical; the rationale behind this new regulation is that the pilots, or in this case the computer systems, on the drone must have the same field of view as real pilots on commercial aircrafts [7]. The integration of radar systems and transponders on aerial systems are, as previously mentioned, astronomically expensive relative to the cost of the aircraft itself, especially for UAVs. The disregard for these systems on smaller aircraft sharing the airspace could prove lethal, since the absence of a transponder ping would cause the aircraft to go undetected.

The limitations of the current technologies lead us to the following research questions:

- Can we develop an aircraft sense and avoid system that does not rely on other aircraft having transponders?
- Can we develop low cost sense and avoid systems for aviation by using cost cutting strategies from the autonomous car industry?
- More specifically, can we replace expensive high fidelity sensors with lower quality

sensors that utilize more complex filters?

## 2 Methods

### 2.1 Design Criteria

The United States Federal Aviation Administration has proposed a series of regulations dictating the conditions under which drones and UAVs flying above 10,000 feet can operate [8]. The administration proposed a minimum *view* distance of 6.4 nautical miles or 12 km. Furthermore, the UAV must have a Field Of View (FOV) of  $220^\circ$  horizontally around the UAV. The sensing system must also have a  $40^\circ$  vertical FOV. Currently, the Traffic Collision and Avoidance Systems, widely used on aircraft, operate accurately at a range of 20 NM; however, the avoidance system only issues Traffic Advisory (TA) when the two airplanes are 3.3 NM away from each other. At this distance, the pilots have 40 seconds before the impending collision, which is sufficient time for pilots to make a avoidance maneuver. The Resolution Action (RA), which demands pilots to take immediate action, is issued when the two aircraft are 2.1 NM away from each other (25 seconds before the collision). The ranges for the TCAS system are illustrated in Figure 3.

One of the main drawbacks of the TCAS system, as previously discussed, is the high cost of the system. Due to this high barrier to entry, generally only large aircraft carry TCAS systems. This leads to the other significant drawback of the system: it requires both parties to have TCAS installed. Aircraft that choose not to install TCAS due to its high cost act as “ghosts” and are undetectable. The goal of the current research is to develop a low cost collision avoidance system that does not rely on transponders for airplane detection while still adhering to the mentioned regulations.

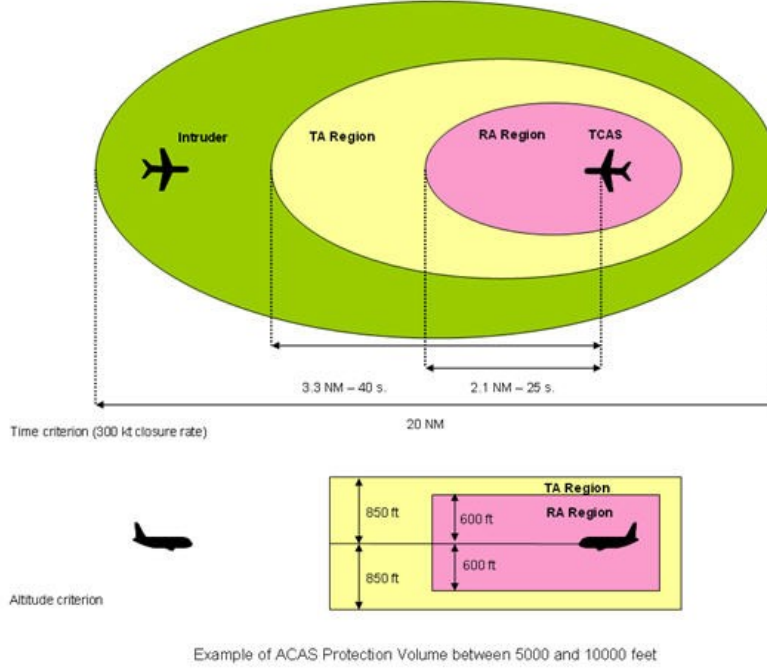


Figure 3: Ranges of TCAS

We decided to develop this new Sense & Avoid system by using the Simrad 4G Radar system. The radar system was chosen because it has a max range of 48 km ( $\approx 26$  NM), which is comparable to max range of TCAS systems. This 26 NM max range would allow the radar system to be deployed in both UAV and Aircraft applications. Furthermore, the Simrad Radar costs only \$2,000, magnitudes less than the previously described cost of TCAS and ADS-B systems. The radar system is also low power (20 W), an essential criteria for UAVs specifically. The radar passes the criteria for Field of View (FOV) as the emitter/receiver system rotates through a full  $360^\circ$  horizontally with a  $25 \pm 20\%$  vertical beam width (Fig. 4). The large vertical beam width combined with the long range of detection, allow the radar to scan a large volume of airspace for projectiles. Finally, the spinning element of the mechanical radar rotates at 48 rpm. In other words, the radar sweeps through 1 revolution every 1.25 seconds [9].



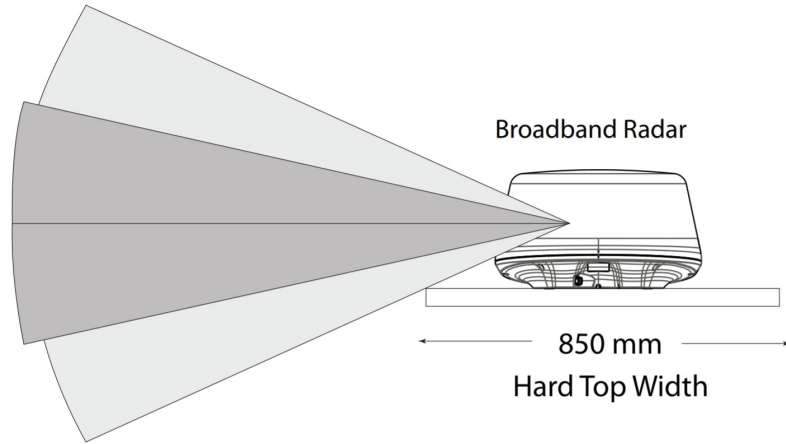


Figure 4: Vertical Angle of Radar

## 2.2 Raw Data Extraction: Radar-Computer Communication

To view the data collected from the Simrad radar, a communications system must be established between the radar and a computer. The Simrad company sells a “black box” system which acts as an intermediary device. However, this device automatically process the RAW radar data using Mini-Automatic Radar Plotting Aid (MARPA) protocols to track objects. Thus, the data sent to the computer is the tracked altered data, not the RAW radar data. This is an issue because the tracking algorithms built into the “black box” were specifically designed for tracking boats, as the radar was built for marine radar applications. The tracking abilities when tracking airplanes are woefully inadequate due to the differences in size and range of objects in marine and aviation applications. Thus, a host of algorithms must be developed to increase the accuracy of the radar tracking in the aviation application. Instead of using the “black box” extension, the radar was connected to a computer by reverse engineering the communications protocol. The numerous wires protruding from the radar were split into two categories: power connections and data connections. The four power connections include two 24V lines (red and yellow) and a ground line (black). The

data connections include 8 wires, which are mapped to the pins of an Ethernet cable. When plugged directly into a computer or through an Ethernet switch, the radar is assigned an IP address on which it will communicate with the computer.

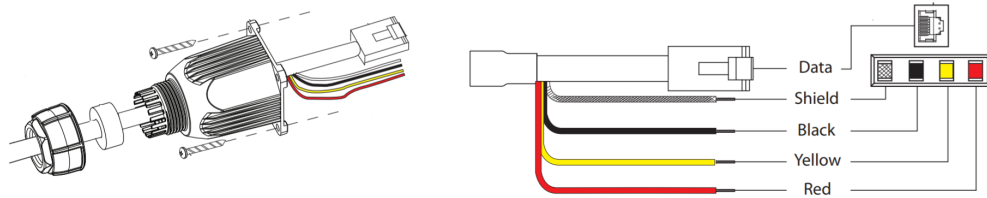


Figure 5: Radar - Computer Connections

To retrieve data from the radar over the Ethernet, multiple C++ scripts were written. The radar defaults to low power mode when it is first connected to the 24V power supply. The radar waits for a specific *wake up* command to be sent over the ethernet port to actually turn the radar on. This data signal was found through reverse engineering the Simrad proprietary communications device. Once the radar system accepts this signal, it automatically sends RAW data; however, one of the issues with the data transmission is that the data is not sent at a constant rate. To solve this issue, a Watchdog Timer is set after the end of each radar transmission. If the computer does not receive the next transmission before the timer expires, the code throws an error. Otherwise, the radar/computer interface functions as expected.

Three separate parameters are passed to the C++ code, which receives the data in binary from the radar: radius, angle, and return strength. These values, represented as a combination of bytes, are converted to the standard integer values with the correct bounds that C++ expects. At each discrete raw angle, as the radar revolves, a new vector of data is sent to the C++ script. The indices of the vector represent the radii while the values represent the strength return. Each new vector being transmitted will be referred to as a “spoke”, since it refers to a “slice” of data from a full circle. The radius ranges from 0 to 511, angle ranges from 0 to 2047, and the return strength ranges from 0 to 255. The data can

be graphed in realtime using a C++ script which converts the polar coordinate values (raw radius and raw angle) and transforming it into Cartesian coordinates. To visualize the third dimension (strength return of the radar signal), the colors of the pixels are varied between black (no return), blue (low return), and red (high return). The computer's mouse scroll wheel control was mapped to the `changeRange()` function. The function transmits a series of binary values that command the radar to change its max range. The max ranges are fixed by the manufacturers of the radar and range between 50 m and 48 km. Note that the number of radius values returned remains constant (512) regardless of the max range since the raw radius is relative. Each individual data value, which is a combination of radius  $r$ , angle  $\theta$ , and strength  $s$ , will be referred to as a pixel  $p$ . The difference in resolution of the pixel at different ranges is illustrated:

$$p = \{r, \theta, s\}$$

The resolution of the radius is defined as the number of pixels per meter:

$$R = \frac{512}{r_{max}}$$

If the max radius is 50m, the resolution is 10.2 pixels/m; however, if the max range is set to 48km, the resolution reduces to 0.01 pixels/m. The typical wingspan of a commercial aircraft is 30 m. By rearranging the equation for resolution, it is evident that the max range of the radar should be set to values less than  $\approx 16$  km for aircraft to appear larger than 1 pixel.

## 2.3 Radar Calibration

To make sure the reverse engineered communications protocol was working, the radar system was installed on the roof of the Aurora Science Building so that the radar system had a full view of the Boston-Cambridge skyline. The data as the radar sweeps through  $360^\circ$  was superimposed on top of the geographic map of the Cambridge/MIT area to gain a physical

meaning from the data.

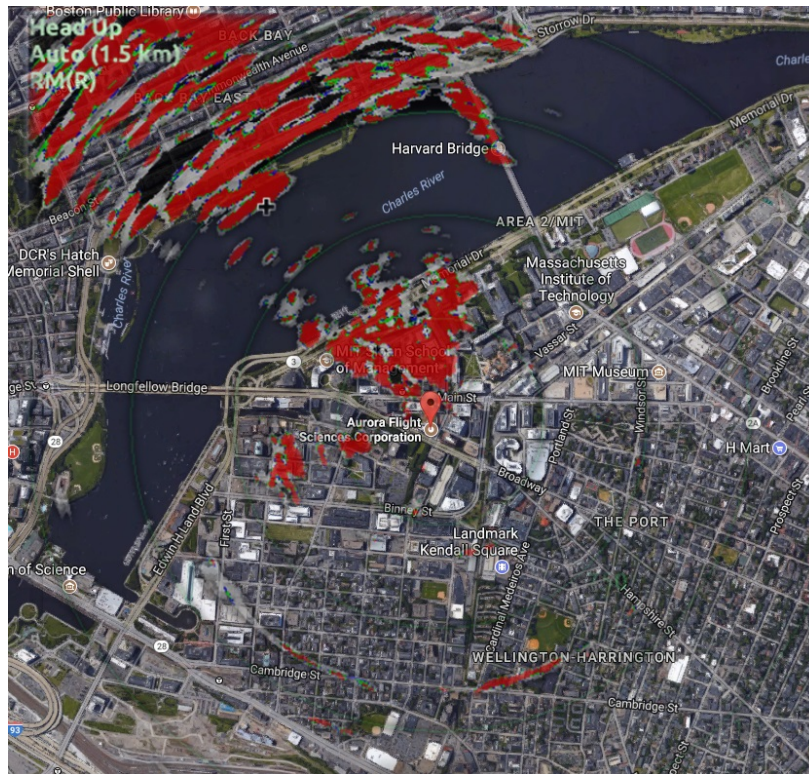


Figure 6: Map with Radar Values

High radar returns, shown in red, are concentrated over the Harvard Bridge and the skyscrapers in Boston. Thus, the superimposed map shows consistency between the radar data and the physical world. The accuracy of the radar system was determined by measuring the length of multiple objects in the radar field of view, such as the Harvard Bridge, and comparing these values with their actual lengths. The percent error of all these measurements were found to be within 1% of the actual length of the objects. Thus, the radar was determined to be properly calibrated.

## 2.4 Development of Memory Optimized Cluster Identification Algorithm

The calibrated radar sends a 2-D matrix of strength returns to the computer. However, this matrix must be processed to actually identify the exact locations of any airplanes that are visible in the radar's field of view. An aircraft will be visible as a group of pixels with non-zero values in the 2-D matrix. We define the word *cluster* as a grouping of pixels that all have non-zero strength returns and are adjacent to at least one other pixel in the group that represent an aircraft.

Modern cluster identification Algorithms import the full 2-D matrix before identifying the clusters. However, it is impractical to load the whole raw data set and then run traditional cluster recognition algorithms, because RAW data is logged into a `.txt` file at the rate of 1Gb every minute. In merely 10 minutes of data capture, the raw data set would consume 10Gb of RAM on a computer. Thus, it is essential that a memory optimized cluster identification algorithm is developed to overcome the shortcomings of the application of traditional cluster recognition algorithms in big data applications. Traditional cluster identification programs also cannot be used because they rely on the fact that all the data is taken at the same time (e.g. camera). The radar, however, transmits data one spoke at a time, meaning that the data at angle 2047 was taken  $\approx 1.25$  seconds before the data at angle 0.

We developed a memory optimized cluster identification algorithm that works by only storing the last two columns of the data set (current and previous spoke). When these two columns are processed, a new spoke from the radar is appended to the 2-D matrix. The new last two columns in the data set (the new current and previous spoke) are processed. Thus, at any given time the computer is only storing two columns of data, reducing the amount of RAM needed to run the script to a mere 20 kB. Thus, the required memory to run the script is reduced by a factor of 50,000.

0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	1	1	0	0	0
0	1	0	1	1	0	0
0	1	1	1	1	0	0
0	0	1	0	0	0	0

Figure 7: Data of Cluster Identification Algorithm

To visualize the processing algorithm, observe Figure 7, which illustrates the full input data in the form of a two dimensional array. Note that the illustration simplifies the data by representing all non-zero values as 1, so that the matrix appears to be a binary *hit* or *miss*. The algorithm begins by looping through all the pixels in a single vertical column and identifying all adjacent non-zero pixels. Each of the concurrent strings of adjacent pixels is stored in a specialized **Cluster** class. This class holds an array of all the pixels, the minimum radius value, the maximum radius value, and the area. The minimum and the maximums of the clusters from the new spoke are compared to the min and max of the previously identified clusters from the last spoke. If the clusters overlap `if(!(min1>max2)|| (min2>max1))`, then the clusters are merged. Any clusters from the previous spoke that are not merged with new clusters have been fully combined and are excluded from the merging process at the next step. When a cluster is finished merging, the center of the cluster is calculated using the following formulas:

$$x_{cent} = \frac{\int x dA}{A}, \quad y_{cent} = \frac{\int y dA}{A},$$

This formula for the center assumes that  $x_{cent}$  is mapped to the RAW angle, and  $y_{cent}$  is mapped to the RAW radius.

Using the mentioned algorithm, data collected from the radar (which was store in a `.txt` file) was automatically post-processed. The input data is graphed as a two dimensional array with the colors representing the return strength of the radar signal (see figure 8).

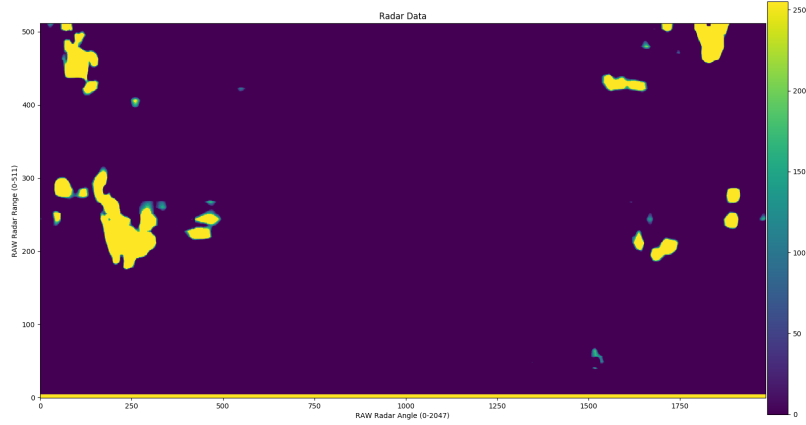


Figure 8: RAW Radar Data as 2-Dimensional Array

Each of the 24 clusters in the data were accurately identified by the algorithm, and the centers (in terms of  $r$  and  $\theta$ ) were recorded.

## 2.5 Tracking and Predicting Clusters over Time

The cluster identification algorithm identifies and returns the position of the aircraft in the field of view at each snapshot in time. To track the aircraft, the position of the aircraft at each of the snapshots, hereafter referred to as *timestep*, must be strung together. If the cluster identification algorithm only identifies one airplane, correlating the motion of of the airplane at each timestep is straightforward – each time the radar sweeps through a complete  $360^\circ$ , we see the aircraft’s new location. However, correlating the positions from the previous and current timesteps is much more difficult when there are multiple aircraft’s identified by the algorithm. In the case that two aircraft pass by each other, the previously mentioned method would fail because it wouldn’t know which one of the two clusters from the previous timestep correlates to the two new clusters. The accurate stringing of the locations of the airplanes at each timestep is called *track association*. We outline two methods for an accurate

track association: a low processing power moderately accurate version and a high processing power accurate version. As the radar system detects local aircraft with each  $360^\circ$  sweep, the position of the aircraft changes significantly. This phenomenon occurs due to the change in time between each sweep. For a standard commercial aircraft with a cruising speed,  $v_{cruising}$  of 900 km/hr, the change in position for an aircraft at cruising speed can be estimated as follows:

$$\Delta t = \frac{1}{T} = \frac{1}{48rev/min} \frac{60sec}{min} = 1.25seconds$$

$$v_{cruising} \approx 900km/hr, \quad \Delta d = v_{cruising} \Delta t = 312.5m$$

Thus, in a single  $360^\circ$  sweep, the aircraft being tracked has traveled more than 6 times its body length. Coupled with the low resolution of the radar system, the position of the aircraft seems to make sudden *jumps* between frames. The significant changes in position due to the discrete time domain is also problematic when the radar system attempts to track multiple objects; as the objects converge, it becomes increasingly difficult to determine where each individual object moved. To overcome this drawback of the lower resolution radar, both methods use a state estimation algorithm that estimate the position of the aircraft at the next radar sweep. The predictions can be compared to the actual locations of aircraft to correlate the data from the current and previous timesteps.

### 2.5.1 Track Association using Euler Approximation

The first method for ensuring an accurate track association involves calculating the position and velocity of each cluster and applying euler approximations to make a prediction of the next position. The position vector, between the radar and tracked aircraft, is defined by  $r$  and  $\theta$ . Similarly, the two velocity vectors are represented in polar coordinates by  $(v_1, \theta_1)$  and  $(v_2, \theta_2)$  respectively. The subscript  $k$  represents the state during the current time step, while the subscript  $k - 1$  represents the state at the previous time step.



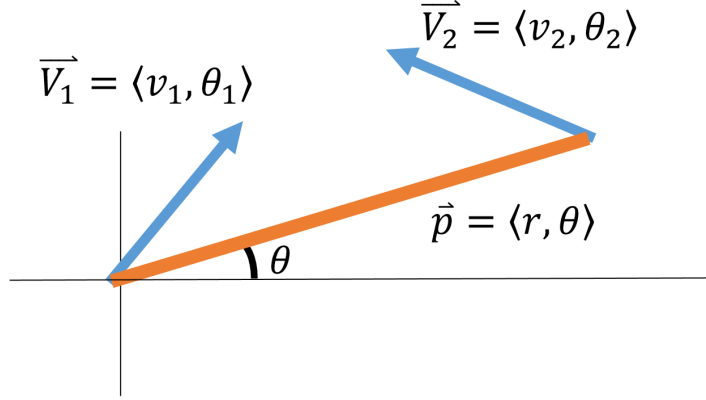


Figure 9: Motion of Two Aircraft in Polar Coordinates

The radius at the next sweep is estimated based on previous known parameters:

$$r = \sqrt{x^2 + y^2}$$

$$\frac{\partial r}{\partial t} = \frac{1}{2}(x^2 + y^2)^{-1/2} * \left( 2x \frac{\partial x}{\partial t} + 2y \frac{\partial y}{\partial t} \right)$$

$$\frac{\partial x}{\partial t} = v_x = v_1 \cos \theta_1 + v_2 \cos \theta_2$$

$$\frac{\partial y}{\partial t} = v_y = v_1 \sin \theta_1 + v_2 \sin \theta_2$$

To calculate  $\dot{r}$ , we next convert from cartesian to polar and substitute the velocities  $v_x$  and  $v_y$ .

$$\dot{r} = \begin{bmatrix} \cos \theta & \sin \theta \end{bmatrix} \cdot \begin{bmatrix} v_1 \cos \theta_1 + v_2 \cos \theta_2 \\ v_1 \sin \theta_1 + v_2 \sin \theta_2 \end{bmatrix}$$

$$\hat{r}_k = \hat{r}_{k-1} + \Delta t \begin{bmatrix} \cos \theta_{k-1} & \sin \theta_{k-1} \end{bmatrix} \cdot \begin{bmatrix} v_{1k-1} \cos \theta_{1k-1} + v_{2k-1} \cos \theta_{2k-1} \\ v_{1k-1} \sin \theta_{1k-1} + v_{2k-1} \sin \theta_{2k-1} \end{bmatrix}$$

The angle after the next sweep is also estimated based on the previous known parameters:

$$\theta = \tan^{-1}(y/x)$$

$$\dot{\theta} = \frac{1}{1 + y^2/x^2} \left( \frac{1}{x} \frac{\partial y}{\partial t} - yx^{-2} \frac{\partial x}{\partial t} \right)$$

$$\dot{\theta} = \frac{1}{x^2 + y^2} \left( -y \frac{\partial x}{\partial t} + x \frac{\partial y}{\partial t} \right)$$

To calculate  $\dot{\theta}$ , we next convert from cartesian to polar and substitute the velocities  $v_x$  and  $v_y$ .

$$\dot{\theta} = \begin{bmatrix} -\sin \theta / r & \cos \theta / r \end{bmatrix} \cdot \begin{bmatrix} v_1 \cos \theta_1 + v_2 \cos \theta_2 \\ v_1 \sin \theta_1 + v_2 \sin \theta_2 \end{bmatrix}$$

$$\begin{bmatrix} -\sin \theta_{k-1} / r_{k-1} & \cos \theta_{k-1} / r_{k-1} \end{bmatrix} \cdot \begin{bmatrix} v_{1k-1} \cos \theta_{1k-1} + v_{2k-1} \cos \theta_{2k-1} \\ v_{1k-1} \sin \theta_{1k-1} + v_{2k-1} \sin \theta_{2k-1} \end{bmatrix}$$

$$\hat{\theta}_k = \hat{\theta}_{k-1} + \Delta t \dot{\theta}_{k-1}$$

### 2.5.2 Track Association using an Extended Kalman Filter

Similar to the prior method, the second method also uses both position and velocity data to ensure an accurate track association. However, the key difference is that the proposed method uses a Kalman Filter to create a more accurate prediction of the location of the aircraft. The refined prediction is created due to the iterative process of the Kalman Filter, which drives the error in prediction to zero. The only drawback to this method is the increase in processing power, due to the iterative process of the Kalman Filter, in order to track objects in real-time.

The implementation of the Kalman filter first transforms the data from the radar, which is represented in a  $r$  and  $\theta$  polar coordinate system, to cartesian coordinates. However, due to the nonlinear nature of the transformation, a standard Kalman Filter cannot be

used. Instead, an Extended Kalman Filter (EKF) is implemented, which linearizes about an estimate of the current mean and covariance in the discrete time domain.

### Polar to Cartesian Conversion for State Matrix

$$\begin{bmatrix} \theta \\ r \end{bmatrix} \longrightarrow \begin{bmatrix} r_k \cos \theta_k \\ r_k \sin \theta_k \\ (r_k \cos \theta_{k-1} - r_{k-1} \cos \theta_{k-1})/\Delta t \\ (r_k \sin \theta_{k-1} - r_{k-1} \sin \theta_{k-1})/\Delta t \end{bmatrix} = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} = \hat{X}_k$$

**Predicting the State Matrix** The prediction of  $\hat{X}$  at the new state is approximated through linearization.

$$\hat{X}_k = A * \hat{X}_{k-1}$$

$$\hat{X}_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \hat{X}_{k-1}$$

The correlation between the position and velocity components of  $\hat{X}$  are captured in the covariance matrix  $P$ . Each element in the matrix,  $P_{ij}$ , represents the degree of correlation between the  $i$ th state variable and the  $j$ th state variable. The covariance matrix is predicted as follows:

$$P_{k|k-1} = AP_{k-1}A^T + WQ_{k-1}W^T$$

**Updating the State Matrix** Based on the predicted state matrix and covariance matrix, the Kalman gain  $K$  is calculated:

$$K = \frac{P_{k|k-1}H_k}{H_kP_{k|k-1}H_k^T + R}$$

Intuitively, the Kalman gain acts as a correction factor used during the prediction phase of each iteration. The Kalman gain is updated in order to drive the residual of the predicted and observed states to zero. The observational model matrix  $H_k$  is defined as the Jacobian of the inputted data.

$$H_k = \frac{\partial h}{\partial x} \Big|_{\hat{x}_{k|k-1}}$$

Since the inputted data is in terms of  $\theta$  and  $r$ , the observational model is found to be:

$$h = \begin{bmatrix} \theta \\ r \end{bmatrix} = \begin{bmatrix} \tan^{-1}(y/x) \\ \sqrt{x^2 + y^2} \end{bmatrix}$$

$$H = \begin{bmatrix} \partial\theta/\partial x & \partial\theta/\partial y \\ \partial r/\partial x & \partial r/\partial y \end{bmatrix} = \begin{bmatrix} \frac{-y}{x^2+y^2} & \frac{x}{x^2+y^2} \\ \frac{x}{\sqrt{x^2+y^2}} & \frac{y}{\sqrt{x^2+y^2}} \end{bmatrix}$$

The states are updated based on both previous predictions and new data regarding the current state. An essential step to this calculations involves determining the residual between the predicted state  $\hat{X}_k$  and the actual state  $z_k$ . The data input at the current state  $k$  is written as:

$$z_k = h(\hat{X}_k) + v_k$$

The measurement of residual is defined as  $\hat{y}_k = z_k - h(\hat{X}_{k|k-1})$ . This residual and Kalman gain are directly related to the change in the values of the state matrix during the update process.

$$\hat{X}_k = \hat{X}_{k-1} + K\hat{y}_k$$

On the other hand, the covariance matrix is updated by considering the Kalman gain and the observational model.

$$P_{k|k} = (I - KH)P_{k|k-1}$$

The prediction and update processes are iterated through at each time step in the discrete time domain. In other words, at the end of each loop  $\hat{X}_k$  becomes  $\hat{X}_{k-1}$  and  $\hat{P}_k$  becomes  $\hat{P}_{k-1}$  based on the properties of recursion. The updated states, which are now considered to originate from state  $k - 1$  instead of  $k$ , now form the input matrix of our next prediction. The outlined Extended Kalman Filter was integrated into the C++ code for the tracking algorithm.

## 2.6 Collection of Truth Data using ADS-B

To determine the accuracy of the radar system, the output data from the Extended Kalman Filter must be compared to some form of *truth data* which includes the actual position and velocity at a given time. This *truth data* is collected using Automatic Dependent Surveillance - Broadcast (ADS-B). The ADS-B technology works by broadcasting information about an aircraft's GPS location, altitude, ground speed and other data to ground stations and other aircraft, approximately once per second. Air traffic controllers and aircraft equipped with ADS-B can immediately receive this broadcasted information [10].

A low cost ADS-B monitoring system operating at 978 MHz was installed at the Aurora Flight Sciences building and used to record the broadcasted data from nearby aircraft as they begin their ascend after taking-off from Boston-Logan International Airport. Multiple scripts written in C were used to handle the ADS-B to USB to computer communications, parse the proprietary data communication protocol, log relevant data to `.txt` files, and graph the positions of the aircraft in realtime on top of a Google map using the Google Maps API.

After the ADS-B is parsed, it must be converted into the same matrix system as the radar. The ADS-B data is originally in the form of Latitude-Longitude-Altitude (LLA); however, the radar data is logged in a North East Down (NED) format. To convert between LLA to NED, an intermediary conversion must occur. Initially the Earth-centered Earth-fixed (ECEF) position is calculated from geodetic latitude, longitude, and altitude (LLA) above

planetary ellipsoid. The formula for this conversion is as follows [13]:

$$\vec{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} r_s \cos \lambda_s \cos t + h \cos \mu \cos t \\ r_s \cos \lambda_s \sin t + h \cos \mu \sin t \\ r_s \sin \lambda_s + h \sin \mu \end{bmatrix}$$

where the geodetic latitude ( $\bar{\mu}$ ), longitude ( $\bar{i}$ ), and altitude ( $\bar{h}$ ) above the planetary ellipsoid into a 3-by-1 vector of ECEF position ( $\bar{p}$ ). The geocentric latitude at mean sea-level and the radius at a surface point ( $r_s$ ) are defined by flattening ( $\bar{f}$ ), and equatorial radius ( $\bar{R}$ ) in the following relationships:

$$\lambda_s = \arctan((1 - f)^2 \tan \mu)$$

$$r_s = \sqrt{\frac{R^2}{1 + (1/(1 - f)^2 - 1) \sin^2 \lambda_s}}$$

The intermediate ECEF format is converted into the desired NED format using the following formula [14]:

$$DCM_{ef} = \begin{bmatrix} -\sin \mu \cos t & -\sin \mu \sin t & \cos \mu \\ -\sin t & \cos t & 0 \\ -\cos \mu \cos t & -\cos \mu \sin t & -\sin \mu \end{bmatrix}$$

such that  $DCM_{ef}$  represents the Direction Cosine Matrix, which captures the rotation about the longitudinal axes and latitudinal axes.

## 3 Low Cost Radar System Testing and Results

### 3.1 Testing of Radar Tracking Algorithm

The State Estimation Algorithm with the Extended Kalman Filter was tested by tracking aircraft in the local airspace near the Boston-Logan International airport. The Simrad 4G radar system was installed on top of the Aurora Flight Sciences Research and Development Office in Cambridge, Massachusetts. The building is located  $\approx 6$  km from the airport. The radar's ability to track the airplanes were determined by multiple non-controllable factors:

- Weather Conditions – the radar cannot be operated outdoors during the rain as it is not properly electrically insulated currently. The solution to this issue is to apply heat-shrink tubing to the solder joints such that the metal is not exposed, allowing it to be water proof. The entry points for all cables externally connected to the radar system (including the power connection and multiple data connections) should be sealed with water-proof rubber connectors.
- Wind Direction – Air Traffic Control (ATC) generally directs airplanes to take-off and land while moving into the wing (so that they do not have any tail wind). The Aurora Flight Sciences Building is located due west of the airport; thus, aircraft cannot be tracked if they approach the airport in the east -west direction. Similarly, if aircraft are forced to take-off in the west to east direction, they will not be able to be tracked as the flight path will never pass by the building.

Although these factors seem to be significant limitations to the Low Cost Radar System technology, they only apply to the case where the radar has a zero velocity near the airport. When installed on an airplane, the weather conditions and wind direction will have no impact on the track itself; these parameters only changed the aircraft's preferred route during ascension or descension.

Radar data was collected on July 28th, 2017. Due to the south-east facing wind direction, during the data collection time frame the wind direction pointed North West facing runway for take-off. As airplane took-off, they curved leftwards in a large circular fashion around the Cambridge city area.

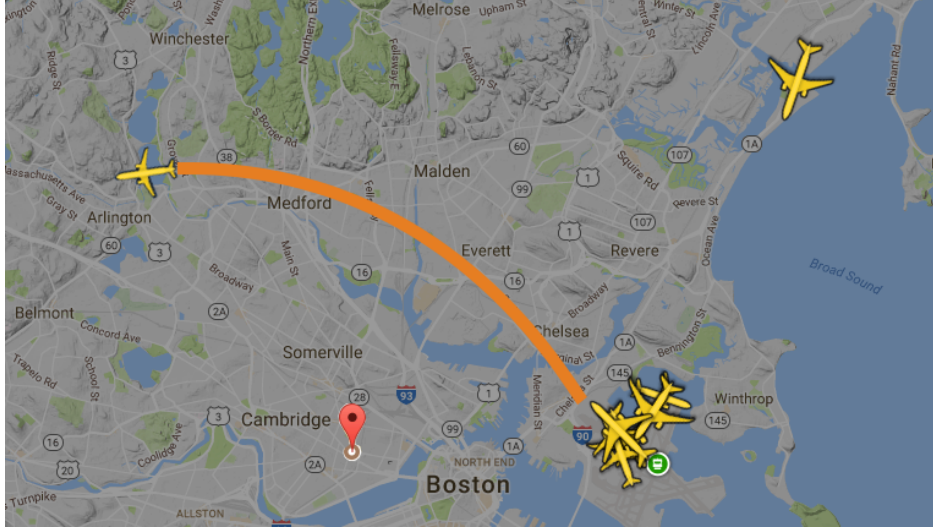


Figure 10: Take-Off Flight Plan from Boston-Logan Airport

The data from the radar was once again unwrapped to form an image for each revolution of the radar angle sweep. It was hypothesized that only a few returns would pick up the airplane due to the relatively far range the radar was operating at along with the radar's low resolution. As demonstrated in Figure 11, the airplane is hardly visible when viewing the whole image. In fact, out of the  $\approx 1,000,000$  returns in each revolution's matrix ( $512 \times 2048$ ), only 5 returns were from the actual airplane. The 2-D matrix at each timestep was subtracted from the previous timestep to locate all the clusters that were moving. This removed all the large yellow stationary spots near the bottom of the matrix which were the tops of buildings in the Boston/Cambridge skyline. In Figure 11, the airplane is the tiny yellow dot near an angle of 300 and a radius of 450. To assist the determination of the location of the airplane, a bright green circle was placed around the airplane's yellow dot, such that the yellow dot



is at the center of the circle. Figure 12 shows the initial image magnified by a factor of 100.

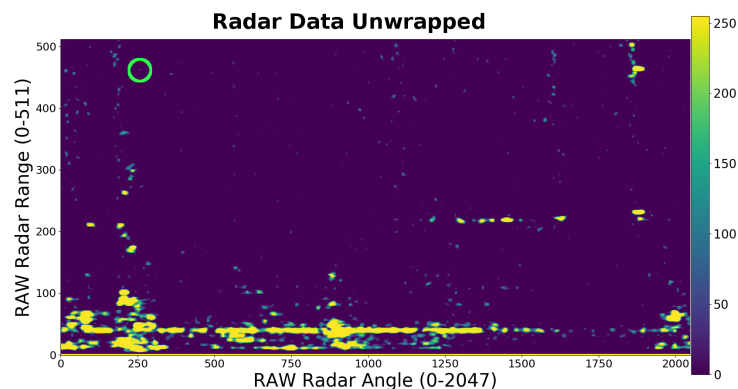


Figure 11: Unwrapped Radar Data for One Revolution – data contains Airbus A320-232 airplane being tracked

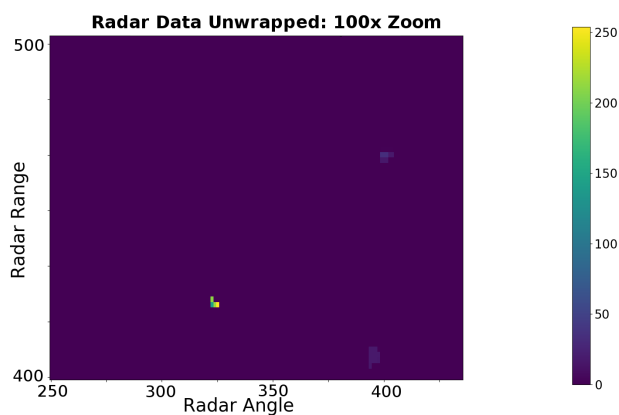


Figure 12: Magnified Image – bright yellow spot near  $r \approx 325$  and  $\theta \approx 425$  is the Airbus A320-232 being tracked

The location of the airplane was tracked for 11 frames as the airplane passed through the field of view of the radar. Although the airplane is difficult to identify from the RAW unwrapped data, the airplane is easily distinguishable when the data is magnified by a factor of 100. Each of the images from the 11 frames were superimposed on top of each other to

display the airplane's position through each of the timesteps (Fig. 13). Each of the positions of the airplane is labeled on the image by the time steps ranging from  $k = 1$  to  $k = 11$ .

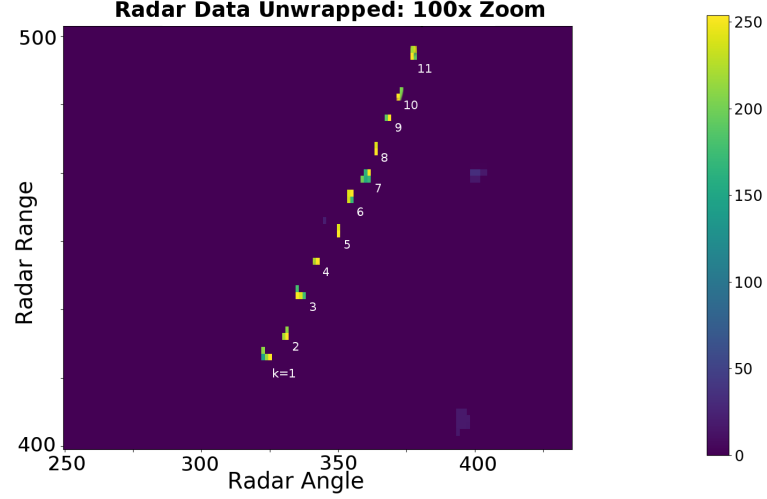


Figure 13: Magnified radar data for each time step overlaid

### 3.2 Determining Accuracy of Airplane Track

The ADS-B parsing algorithm returns the ADS hex code, which acts as an aircraft's unique signature, epoch time, radius  $r$ , angle in the horizontal plane  $\theta$ , and angle in the vertical plane  $\phi$ . Each of the spherical coordinate values is converted into the raw data format that the radar outputs. The conversions are as follows:

$$r_{RAW} = 512 \frac{r}{8000}, \quad \theta_{RAW} = 2048 \frac{\theta}{360}, \quad \phi_{RAW} = 2048 \frac{\phi}{360}$$

Since both ADS-B data and Radar cluster position data are now using the same units, both data sets can be graphed against each other with the ADS-B data acting as the truth data. As the radar collects the strength return data, the Extended Kalman Filter algorithm sifts through all the identified clusters and noise, and outputs the exact location of the

aircraft. The similarity/difference between the position data provided by the EKF and the truth data can be numerically identified by calculating the root mean squared error (RMSE). The RMSE measures the square-root of the average error squared between the EKF data and ADS-B data. We define the actual radius of the object from the ADS-B sensor as  $r_i$  and the radius value from the radar sensor as  $\hat{r}_i$ . The Root Mean Squared Error is calculated as follows:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{r}_i - r_i)^2}$$

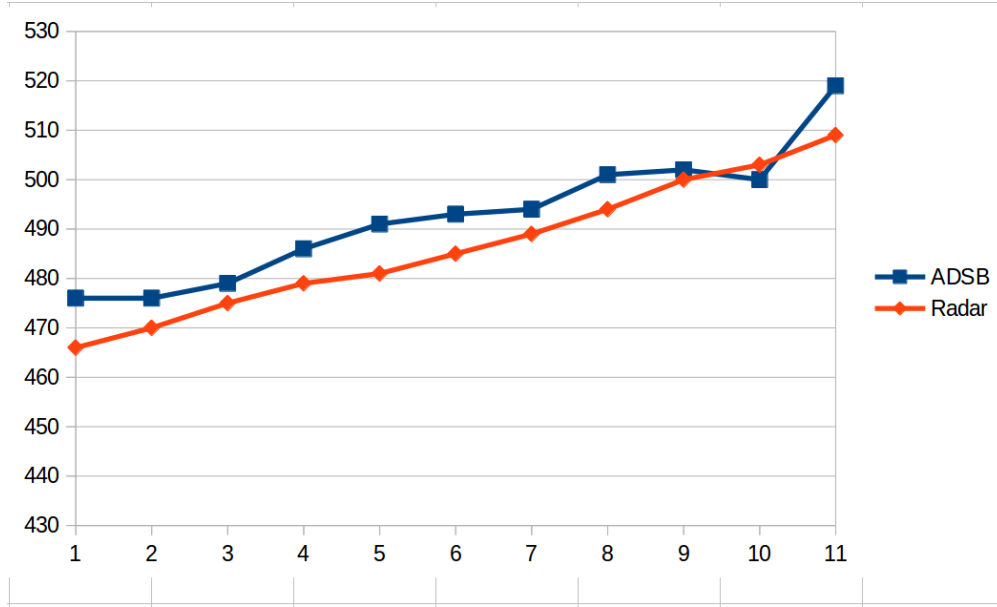


Figure 14: ADS-B Truth Data and Radar Data

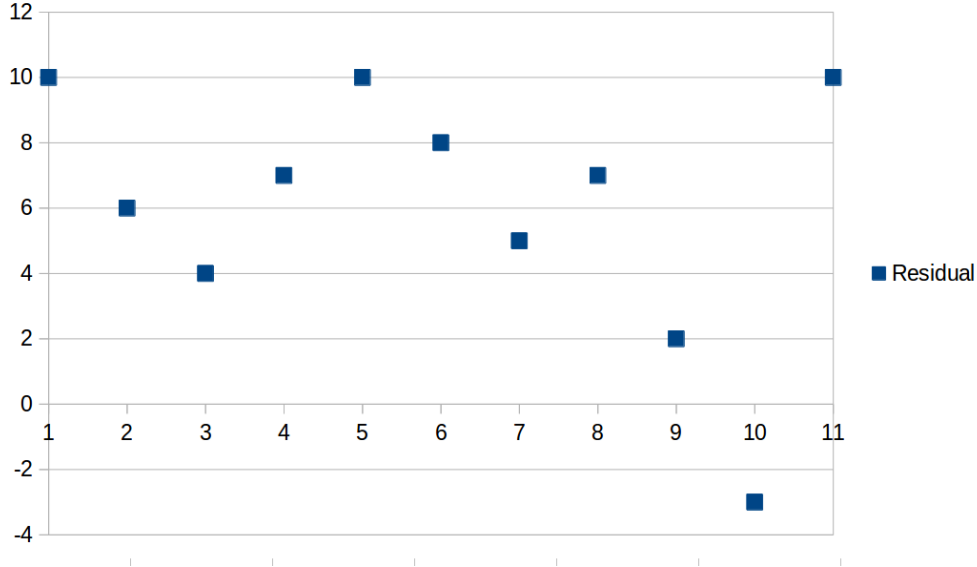


Figure 15: ADS-B v. Radar Residual Plot

Using the mentioned formula for root mean squared error, the  $RMSE$  was calculated to be 7.084. Theoretically, the RMSE should tend to zero when the radar measurements are accurate relative to the ADS-B data. Errors in the radar measurements could include inaccurate calibration of the 9-axis heading sensor, inaccuracies in the initial GPS coordinate of the radar system, and inaccuracies due to low radar resolution. The error is clearly visible in the residual plot, which shows the residuals being skewed to the positive end rather than being centered at zero. This means that the radar consistently underestimates the position. If the zero error was corrected, the RMSE would drop to 3.77, showing a mere 0.5% error in aircraft position measurement. Thus, the Extended Kalman Filter can provide an accurate track association of an aircraft.

## 4 Conclusion

Modern collision avoidance systems for aircraft have been historically astronomically expensive, motivating many smaller aircraft to not use collision avoidance systems. Refraining

from using the collision avoidance systems is dangerous for both the small ghost aircraft and the large commercial aircraft, since the commercial aircrafts cannot detect and avoid the smaller aircraft when they do not have transponders installed. A low cost radar-based collision avoidance system was developed such that any flying hazardous projectile can be tracked and avoided. The tracking system was developed using memory efficient cluster identification algorithms to identify nearby aircraft, state estimation equations to develop track associations to differentiate between multiple nearby moving aircraft, and Extended Kalman Filters for aircraft position and velocity estimation refinements. Initial radar calibration techniques allowed the radar to accurately measure the distance to an object with 1% accuracy. Aircraft taking-off from Boston-Logan Airport were accurately tracked by the developed object tracking algorithms. Furthermore, position data from the radar system was compared to the true values from the ADS-B. Although, the airplane was tracked accurately, the root mean square error between the two data sets was 7.08. Thus, there was a slight inconsistency between the ADS-B data and the radar system, which was most likely caused by inaccuracies in the manually selected initial GPS frame of reference. Applying a zero-error correction results in an RMSE of 3.77, showing a mere 0.5% error in aircraft position measurement. Thus, the low-cost radar system, combined with cluster identification algorithm and Extended Kalman Filter, can accurately track an aircraft.

## 5 Future Work

One of the limitations of the developed radar system is the low resolution, which means that only a few of the 1 million returns contains data of the airplane. The small size of radar returns from the airplane did not significantly reduce the accuracy of the tracking algorithm since sophisticated state estimation algorithms (for cluster prediction and measurement refinements) were developed. However, clusters had to be manually selected before they were

tracked, similar to MARPA tracking; the manual selection was tedious since the cluster is much smaller than the full data set. To overcome this limitation, a machine learning approach can be implemented to determine “points of interest” and automatically begin the tracking algorithm.

The main benefit of using the radar based system is that it does not need nearby aircraft to have compatible transponder systems already installed. However, the use of a radar system also has a few drawbacks. As previously described, radar measurements were used to estimate the other aircraft’s radius and angle in polar coordinates (i.e. in two-dimensions). The wide vertical angle of the radar system, with no vertical resolution, means that the system would detect airplanes that are a safe altitude difference away. Thus, it is essential that objects are tracked accurately in all three dimensions rather than “flattening” the data into two dimensions. To combat this issue, multiple radar systems can be mounted at different orientations. Data from each of these radar sensors can be stitched together to create one complete three-dimensional map. Distance measurements can also be refined by using other low-cost sensors such as LIDAR systems or camera-based visual tracking.

## 6 Acknowledgments

I would like to thank my mentors Jae-Woo Choi, Franklin Wu, and Michael Sardonini for their valuable advice and guidance. I would also like to thank Aurora Flight Sciences for providing the resources necessary for completing this research project. Moreover, I would like to thank Jenny “The Great” for helping make structural changes to the paper and presentation. I would like to extend my gratitude to Elijah Stanger Jones and Shawn Chao for providing valuable comments during the paper review process. I would like to thank my sponsors Ms. Carol Maxwell, Dr. and Mrs. Zheng, Mr. Zhiang Deng and Ms. Xin Chen, and Mr. Charles E. Sane. Finally, I would like to thank the Center for Excellence in Education

for allowing me to conduct research at Massachusetts Institute of Technology and Aurora Flight Sciences as part of the Research Science Institute (RSI) program.

## References

- [1] A new approach to managing congestion in the skies. *Volpe National Transportation Systems Center*, Apr 2015.
- [2] K. I. Shetty and R. J. Hansman. Current and historical trends in general aviation in the united states. *MIT International Center for Air Transportation (ICAT)*, Aug 2012.
- [3] R. Amadeo. Googles waymo invests in lidar technology, cuts costs by 90 percent. *Ars Technica*, Jan 2017.
- [4] J. R. Ramsey. The future of air traffic control, Jul 2008.
- [5] New better-than-radar technology will boost aircraft tracking. *Tech Bit*.
- [6] e. a. Viquerat A. Reactive collision avoidance for unmanned aerial vehicles using doppler radar. *Springer Tracts in Advanced Robotics Field and Service Robotics*, Dec 2007.
- [7] e. a. Viquerat A. Reactive collision avoidance for unmanned aerial vehicles using doppler radar, 2008.
- [8] L. Blain. Echodyne’s amazing miniature uav radar: A huge step towards legal, autonomous drone deliveries, Nov 2016.
- [9] Broadband 4g radar specification sheet. *Simrad*.
- [10] FAA: ADS-B guidelines. *Federal Aviation Administration*, Feb 2017.
- [11] FAA: ADS-B regulation proposal. *Federal Aviation Administration*.
- [12] FAA airspace regulations handbook. *Federal Aviation Administration*.
- [13] Calculate earth-centered earth-fixed (ecef) position from geodetic latitude longitude and altitude above planetary ellipsoid. *Simulink*.
- [14] Convert geodetic latitude and longitude to direction cosine matrix. *Simulink*.