Spatial Autocorrelation for Automatic Reconstruction

Hung Doan

November 13, 2020

Abstract

Automatic reconstruction has been an ongoing problem for decades. Current methods, such as nearest neighbor approaches of automatic reconstruction only obtain about 95% accuracy. New methods such as spatial autocorrelation and machine learning are required to achieve automatic reconstruction with over 99% accuracy. In this paper, we will discuss the various problems when it comes to automatic reconstruction. We focus on spatial autocorrelation as a method for associating radar tracks to GPS objects. Specifically, we use the Moran's I statistic which uses a spatial weights matrix to define neighbors to be correlated. We find that this method works well for long living tracks and is able to accurately correlate these tracks to GPS data. However, with shorter tracks, the results are not as concise and other methods may have to be used for these short lived tracks. We use the results from the Moran's I statistic as predictors to feed in to MATLAB's classification learner to determine if tracks are correlated or not. However, do to lack of a sufficient data set, the model produced was not accurate enough to use for automatic reconstruction.

1 Introduction

One method of assessing the performance of a radar used to track aircraft is to compare the data generated from the radar to GPS data. This is done by having an aircraft to fly in a test range and allowing the radar to track the aircraft for a given period of time while the GPS pod, fixed to the aircraft, records data. After the testing period, we compare the data generated from the radar to the GPS data generated by the pod.

We consider the GPS data as ”truth” data. Because the pod is fixed directly to the aircraft, the data gathered from the pod can have the name of the object directly tied to the data. For example, if pod is fixed to a Lear aircraft, then Lear is associated with each data point. In this paper, for both GPS data and radar data, each data point contains a time stamp, latitude, longitude, altitude, and velocity in the x, y and z directions.

While it is possible for the object name to be associated to GPS data, it is not possible for the object name to be associated to track data in real time. This is because radars receive updates containing range, bearing, and elevation information. Radars can see objects in the sky but they cannot assess exactly what the object is. Best guesses can be made using radar cross section and other attributes such as speed, but in general, it is not possible to assess the type of aircraft. Instead of associating an object name to the radar update, radars can associate track numbers if they are sophisticated enough. They are able to associate a collection of radar updates to a single track.

We wish to automatically associate track numbers generated from radar data to objects associated with GPS data. We will do this by comparing track kinematics including latitude, longitude, altitude, and velocity in the x, y and z directions. In chapter 2, we will describe the different ways radar data can be associated to GPS data. In chapter 3, we will provide a visualization of the problem. In chapter 4, we will elaborate on the methods used for correlating radar data to GPS data. We will be using spatial autocorrelation to associate the two data sets. We review some theory behind this method and provide results obtained using this method. In chapter 5, we will detail how using MATLAB’s classification learner is a possible solution for automatic reconstruction.

Both GPS data and radar data used in this paper are from the United States Navy. Because both data sets are greater than 25 years old, they are unclassified and were permitted to be used for this paper.

2 Types of Associations

2.1 Track Association

Track association is the simplest form of associating radar data to truth data. There are many ways of associating track data to GPS data and it can become ambiguous with closely spaced objects. In many cases, it can appear as a trivial task to manually associate radar data to GPS data due to how well we are able to recognize patterns. However, the task can be much more complex for a computer.

In Figure 1, we see a picture with two tracks. The two black lines represent GPS (truth) data, and the colored lines represent radar data. For us, it is easy to tell that track 1234 is associated with F/A-18 and track 5678 is associated with Lear, but for a computer it can be a little more complicated. In this case, it is clear that track 1234 is more closely spaced to F/A-18 and track 5678 is more closely spaced to Lear. However, there can be cases when the more closely spaced track is not the proper association.

2.2 Dual Tracks

Sometimes radars can produce multiple tracks on one object. If there already exists a radar track on an object in the sky and another track is created which is tracking that same object, then the new track created is called a dual track. In Figure 2, we can see two objects in the track picture. Track 1234 is associated to F/A-18 with track 1235 as a dual track to the same object. In addition, we have track 5678 associated to Lear. In this paper, we will be using spatial autocorrelation and the method for a associating a dual track will be the same as for the method of associating a primary track (the first track associated). When it comes to auto reconstruction, logic will have to be implemented to distinguish between dual and primary tracks.

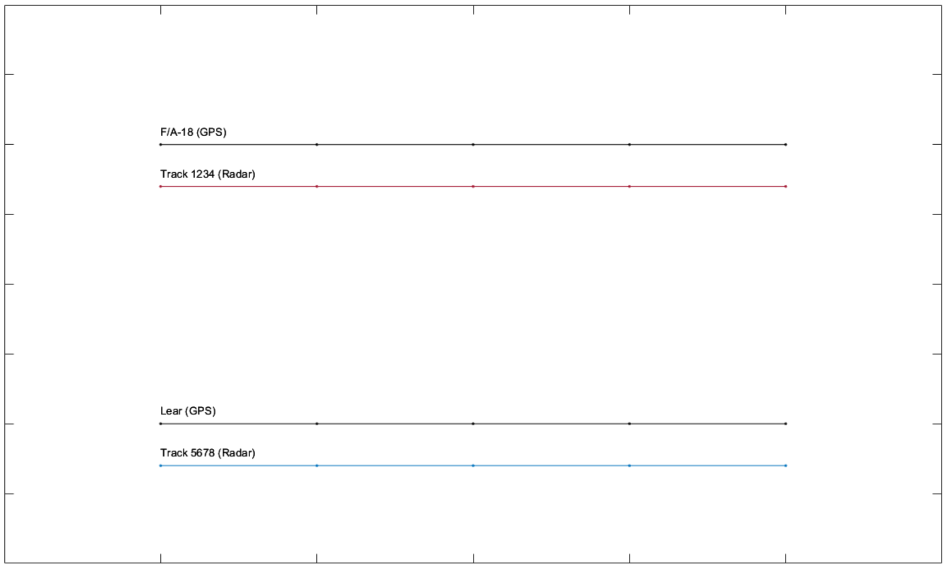


Figure 1: Track 1234 associated with F/A-18 and Track 5678 associated with Lear

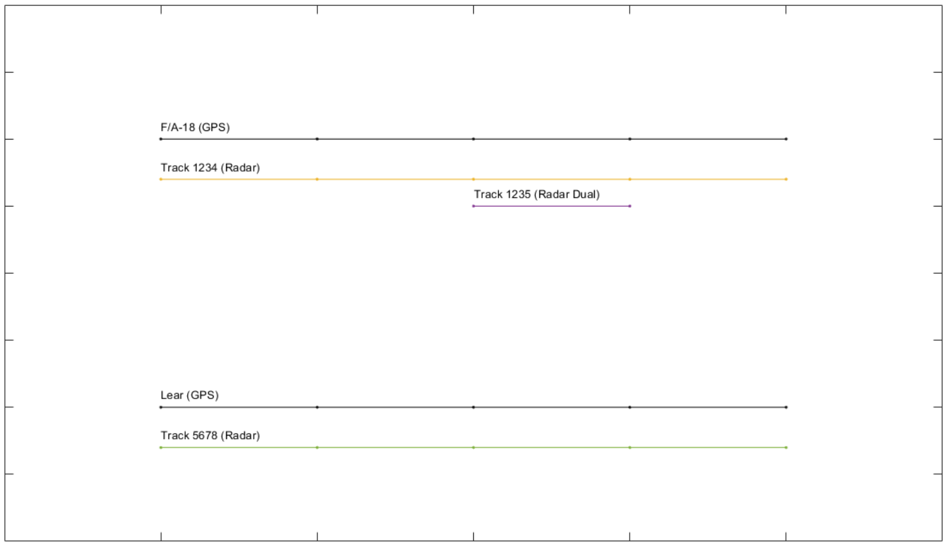


Figure 2: Track 1235 is a dual of F/A-18

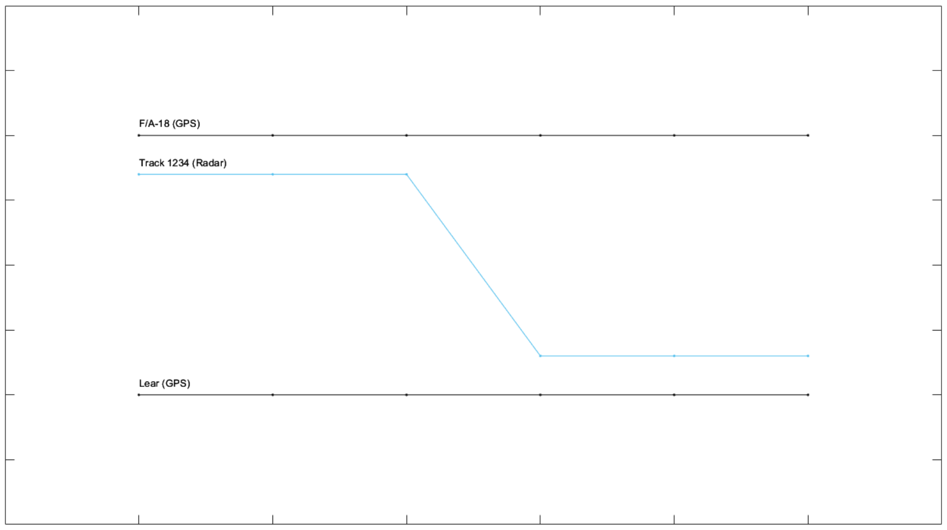


Figure 3: Track 1234 is associated to F/A-18 then becomes associated to Lear

2.3 Track Fracture

A track fracture occurs when a radar is tracking one object and it starts tracking another object using the same track number. The radar does not recognize that the object it was tracking is actually two different objects and has maintained the same track number on two different objects. In Figure 3, we see track 1234 initially associated to F/A-18. After three units of time, track 1234 begins associating to Lear without changing track numbers. To the radar, track 1234 has been associated to the same object the entire time, but in reality, it was initially tracking F/A-18 and started tracking Lear. In this paper, we will not concern ourselves with track fractures using spatial autocorrelation.

2.4 Track Swap

When there are two objects that are closely spaced, it can be difficult for a radar to distinguish if there are actually two objects there. Even if it can, it must determine which radar updates go to which object. Sometimes, the radar may swap tracks. This occurs when a two track numbers are associated to their respectively objects and then start associating to objects

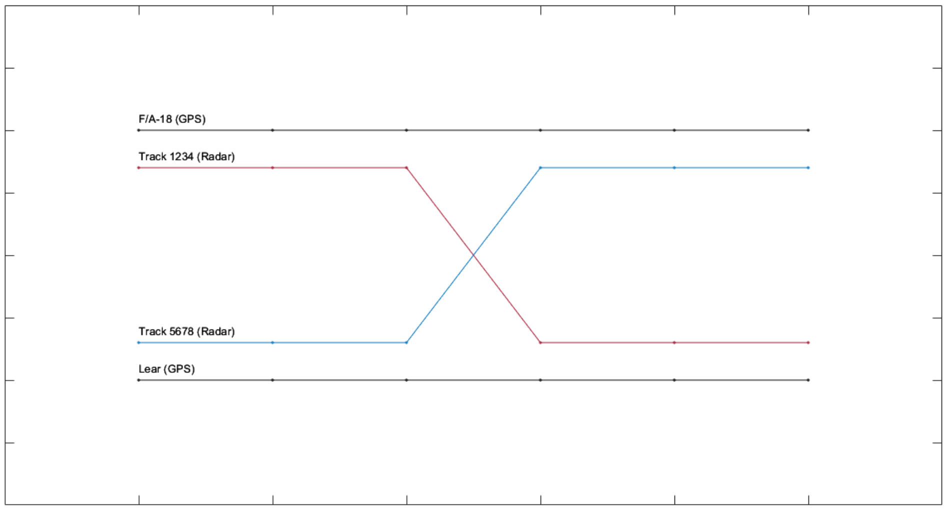


Figure 4: Track 1234 is associated to F/A-18 then becomes associated to Lear while track 5678 is associated to Lear then becomes associated to F/A-18

they were not initially associated to. In Figure 4, we see that track 1234 initially associated to F/A-18 and track 5678 associated to Lear. After three units of time, track 1234 starts associating with Lear and track 5678 starts associating to F/A-18, simultaneously. In this paper, we will not concern ourseleves with track swaps using spatial autocorrelation.

3 Visualization of the Problem

Both the radar and GPS data used in this paper are from the United States Navy. The GPS data contains four objects: Lear 1, Lear 2, CAP 1, and CAP 2. The radar data contains tracks following these four objects and tracks of four ships and a missile. Figure 5 contains the four GPS tracks along with 31 tracks created by the radar. Note that one GPS track can be associated to multiple radar tracks.

Because the track picture is so convoluted, we will narrow down the track picture to just one GPS object, Lear 1, and a handful of possibly associated tracks to Lear 1. In Figure 6, we can see the GPS track of Lear 1 in black, while the other radar tracks are in color and

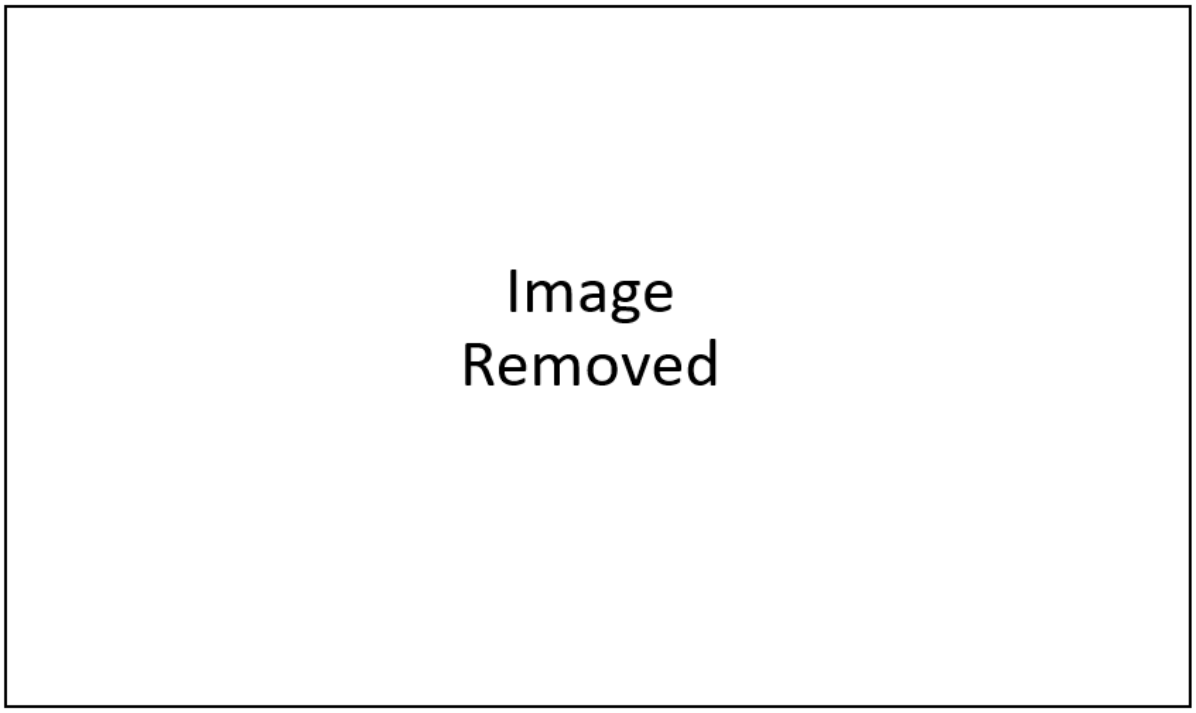


Figure 5: Track picture including both radar and GPS data for all tracks

have track numbers. At first glance at the picture, it appears that track 3 follows Lear 1 very well. We can see that track 632 and track 353 are possible dual tracks of track 3. Track 1347 appears to be a short lived primary track. Track 525 follows Lear 1 very well. Track 555 is a potential candidate that may be associated to Lear 1, though it does not appear to follow the Lear 1 track as well as the other tracks in the picture. And lastly, track 1011 is a track of a missile heading towards Lear 1.

4 Solving the Problem

4.1 Steps taken to solve the problem

As we mentioned in the introduction, radar data typically comes in range, bearing and elevation. However, for ease of analysis, range, bearing, and elevation are converted to WGS latitude, longitude, altitude coordinates. However, because we plan on doing spatial autocorrelation, we convert latitude, longitude, and altitude data to earth centered, earth

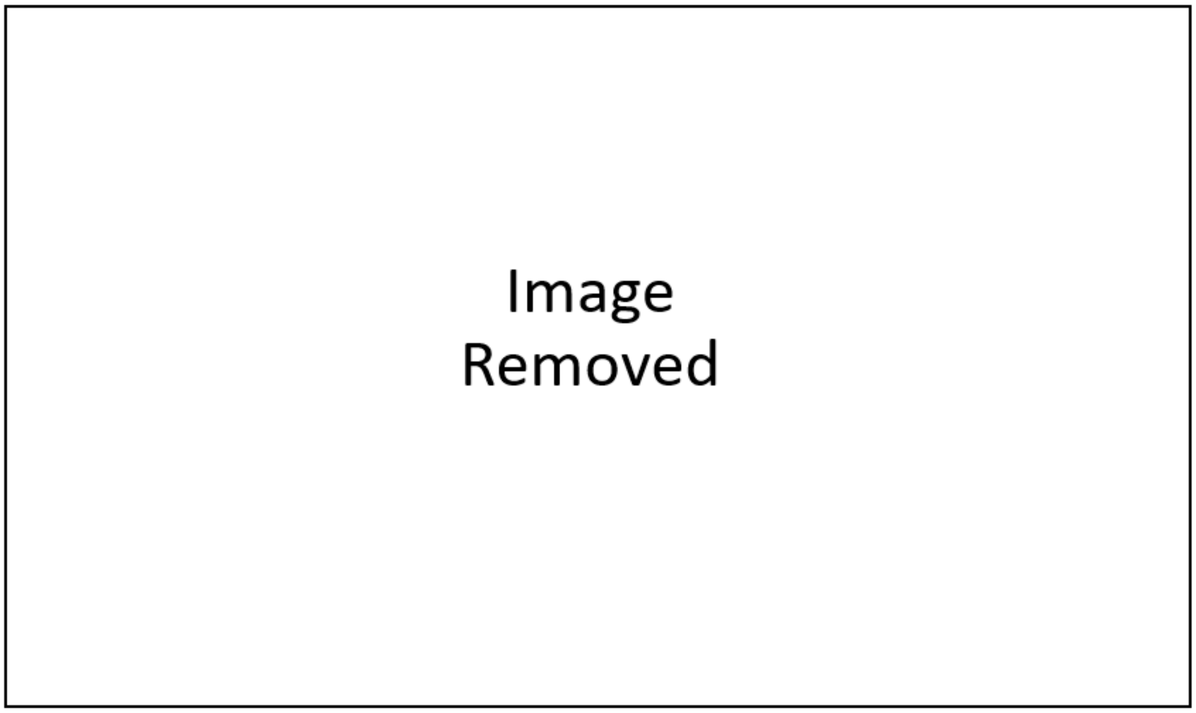


Figure 6: Track picture including both radar and GPS data for Lear 1

fixed (ECEF) x, y, and z coordinates. In addition to these kinematic positions, we also use velocity in the x, y, and z directions.

In order to fair correlate each data point, we first interpolate the data in order to 'time align' the data points. Because radar data points and GPS data points are not recorded at the exact same moment in time, it would be unfair to do correlations if the data points were not exactly time aligned.

Next, we use the Moran's I statistic to determine correlation with each radar track to GPS data. We will be doing correlations on x, y, z, v_x, and v_y kinematics between GPS data and radar data. In this chapter, we will use track 3 and track 555 from Figure 6 as an example of the results obtained from the Moran's I.

Finally, we will use MATLAB's classification learner with the correlations obtained from the Moran's I as the predictors and whether or not the track is correlated as the response. To do this, we will run a comparison of every track from the radar data to every track from GPS data and use this information to feed the classification learner. Because we have four

GPS objects and 31 radar tracks, there were be 124 Morna's I values for x, y, z, v_x and v_y.

4.2 Spacial Autocorrelation

Spatial autocorrelation is the statistic that measures spacial dependence. Eachn statistic of autocorrelation is constructed on the basis of a crossed product between a similarity index, c_{ij}, of the values taken by a variable of interest y and the values taken by the observations of the neighborhood [1]. The general form of spatial autocorrelation is given by

$$\Gamma = \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} \times c_{ij} \tag{1}$$

where w_{ij} is an element of the spatial weights matrix W, $c_{ij} = (y_i - \bar{y})(y_j - \bar{y})$ (Moran's I), y_i is the vector of interpolated values from GPS data, and y_j is the vector of interpolated values from radar data. Plugging in the value of c_{ij} we can obtain the Moran's I statistic given by

$$I = \frac{N}{\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij}} \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij}(y_i - \bar{y})(y_j - \bar{y})}{\sum_{i=1}^{N} (y_i - \bar{y})^2} \tag{2}$$

where N represent the total number of observations, $\bar{y}$ is the arithmetic average o fthe values taken by y, and w_{ij} is an element of the spatial weights matrix W.

4.3 Defining the Spatial Weights Matrix

The spatial weights matrix defines who neighbors are for spatial autocorrelation. Because we set up our data in a particular way, we can correlate the ith entry in the GPS data y_i to the jth entry in radar data y_j where $i = j$. Because we time aligned our data, we can compare each point in GPS data to their respective, time aligned point, in radar data. Knowing this, we can separate the spatial weights matrix into four compartments. The following provides a break down of how we will define our spatial weights matrix:

$$W = \begin{bmatrix} A & B \\ C & D \end{bmatrix}. \tag{3}$$

where A is the spatial weights matrix for y_i correlated to itself. This is the zero matrix since we do not wish to correlate GPS data with itself. B is the spatial weights matrix for y_i correlated to y_j. This is the identity matrix since we let the ith element in y_i be a neighbor of the jth element in y_j. C is the spatial weights matrix for y_j correlated to y_i. This is the identity matrix since we let the jth element in y_j be a neighbor of the ith element in y_i. D is the spatial weights matrix for y_j correlated to itself. This is the zero matrix since we do not wish to correlate radar data with itself.

As an example, suppose we want to correlate a GPS track y_i to a radar track y_j. If y_i had four interpolated points and y_j had four interpolated points, then the spatial weights matrix would have the form

$$W = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Note how compartment A is the zero matrix, compartment B is the identity matrix, compartment C is the identity matrix, and compartment D is the zero matrix.

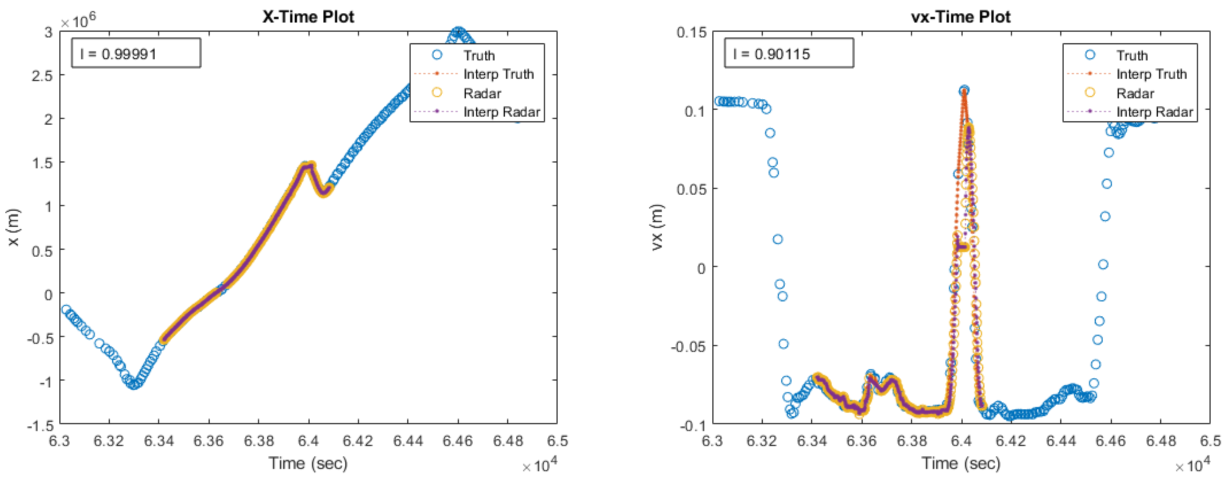
4.4 Results for Spatial Autocorrelation

As an example, we will focus on track 3 and track 555 from Figure 6. We can see just by looking that track 3 is very well correlated with Lear 1 while track 555 seems poorly correlated, if at all correlated. We run the Moran's I statistic on x, y, z, v_x, v_y, and v_z for track 3 and x, y, v_x, and v_y for track 555.

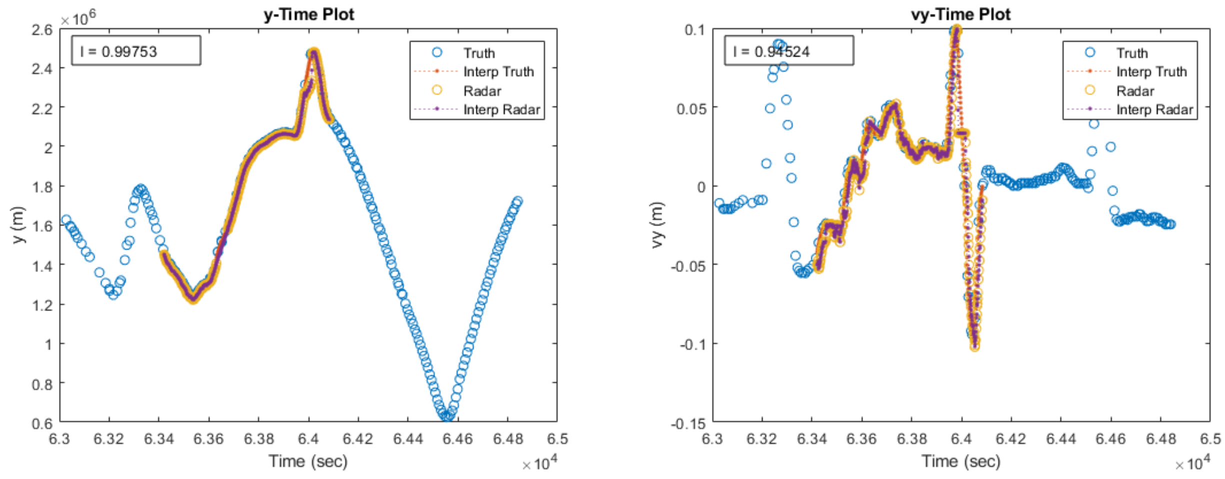
In Figure 7, we see three sub figures. In Figure 7 (a), we have the results for x and v_x. In Figure 7 (b), we have results for y and v_y. And in 7 (c), we have results for z and v_z. The blue circles represent the truth data. The red dashed lines represent the interpolated truth data. The yellow circles represent the radar data. And the purple dashed lines represent the interpolated radar data. Notice how the interpolation only occurs when there exists data for both data sets. We can also see that the truth data spans a larger time than track 3. This can be seen in Figure 6.

The results are as expected. The Moran's I statistic for x and v_x are $I_x = 0.9999$ and $I_{v_x} = 0.9012$, respectively. The Moran's I statistic for y and v_y are $I_y = 0.9975$ and $I_{v_y} = 0.9452$, respectively. And the Moran's I statistic for z and v_z are $I_z = 0.9981$ and $I_{v_z} = -0.0019$, respectively. In Figure 7 (c), we see that the truth data appears to be 0 throughout the entire data set. Certainly, the results for correlating v_z will not be used but they were included here to show that the Moran's I statistic showed no correlation with $I_{v_x} = -0.0019$, as we would expect.

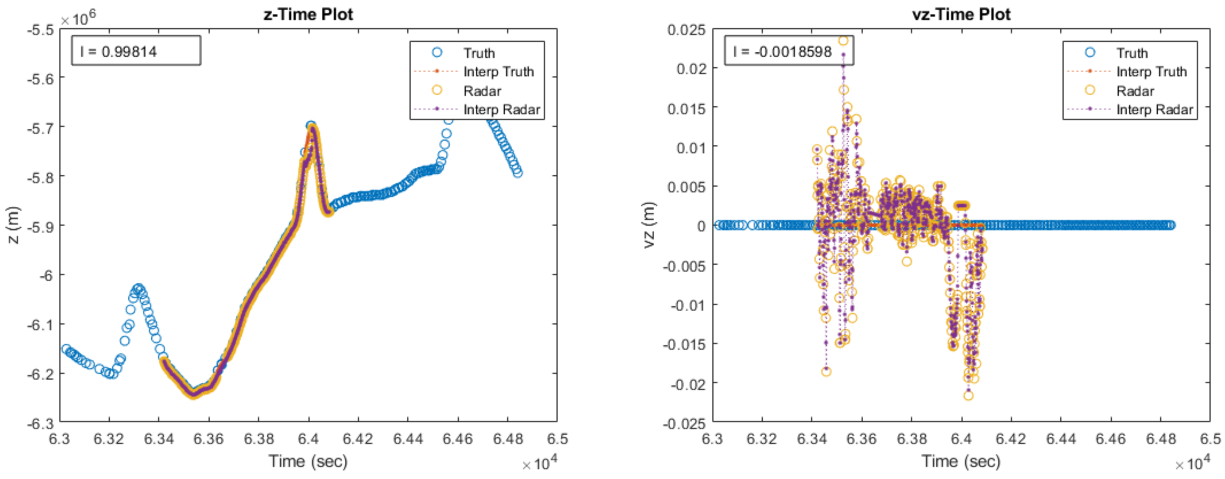
Doing the same analysis, but now on track 555, we obtain the results seen in Figure 8. This time, because v_z cannot be used, we will only calculate the Moran's I statistic on x, y, v_x, and v_y. In Figure 8 (a), we see that the correlation for x seems to be a bit off. This is expected because the track position in latitude and longitude from Figure 6 seemed to be off as well. The Moran's I for the x position was $I_x = -0.1386$. This shows little to no correlation. On the other hand, v_x showed very promising correlation with $I_{v_x} = 0.9887$. This may help with automatic reconstruction because even though the x position is weakly correlated, we can still find strong correlation with v_x. The same is true for y and v_y. The



(a)

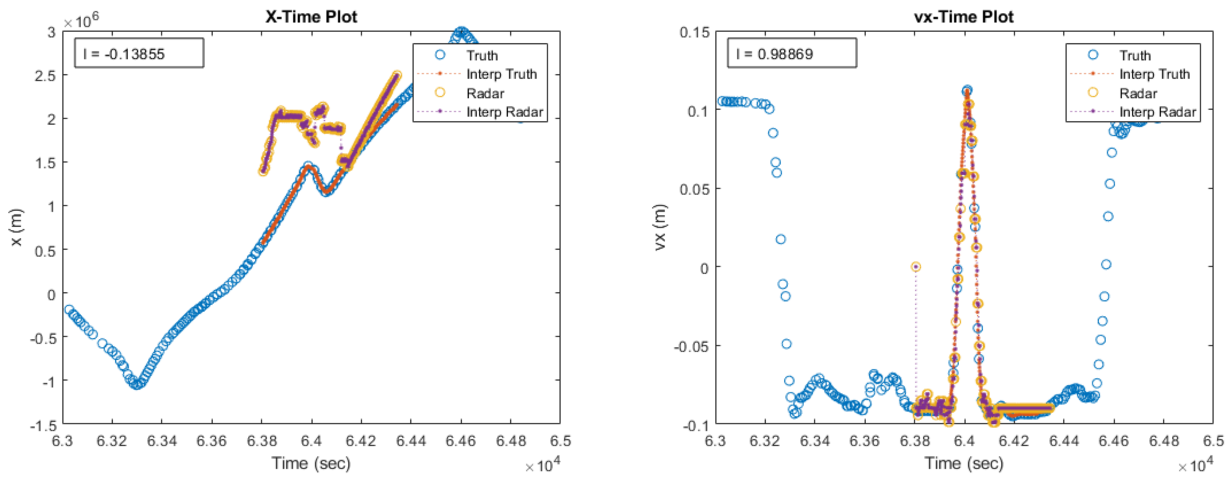


(b)

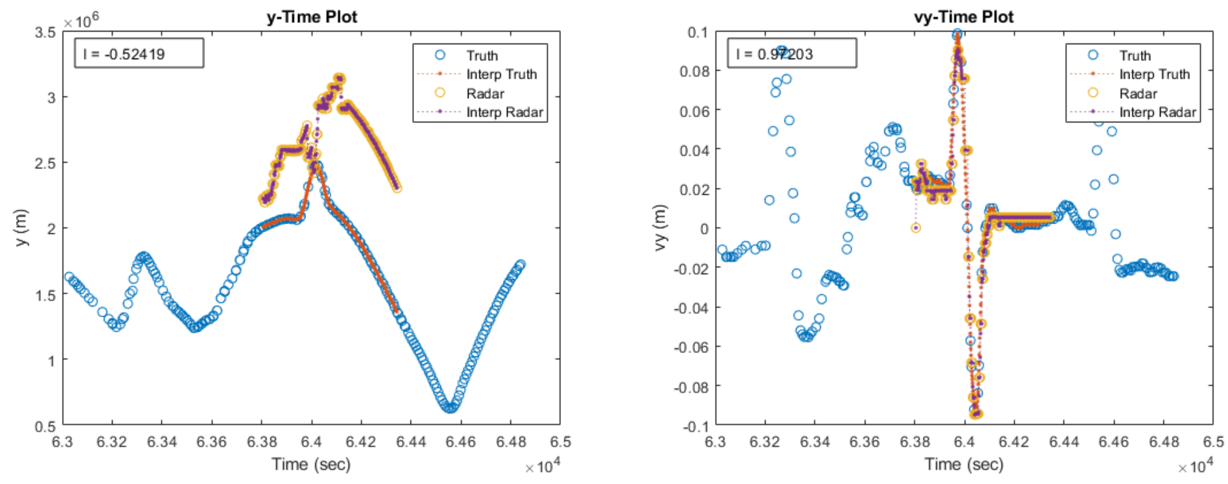


(c)

Figure 7: (a) Results for Moran's I statistic for correlating x and v_x on track 3 (b) Results for Moran's I statistic for correlating y and v_y on track 3 (c) Results for Moran's I statistic for correlating z and v_z on track 3



(a)



(b)

Figure 8: (a) Results for Moran's I statistic for correlating x and v_x on track 555 (b) Results for Moran's I statistic for correlating y and v_y on track 555

Moran's I for y and v_y are $I_y = -0.5242$ and $I_{v_y} = 0.9720$. Again, there is less of a correlation in the y position but the correlation for v_y is very high.

5 MATLAB's Classification Learner

In the previous section, we focused on correlation just two tracks, track 3 and track 555, to one object, Lear 1. Now, we will correlate every radar track to every truth object in the track picture. Because there are 4 truth objects and 31 radar tracks, there are a total of

$4 \times 31 = 124$ different combinations of associations that can be made. We calculate the Moran's I for x, y, z, v_x and v_y, for all 124 combinations. We use these results as predictors to feed in to MATLAB's classification learner so that it may recognize some sort of pattern based on the Moran's I results to call each pairwise association, correlated or not correlated. The results can be seen in Figure **??**. We can see that out of the 124 possible combinations, it correctly classified 107 as correlated or not correlated. There were 12 false negatives and 5 false positives. The accuracy of the model made by MATLAB was 86.3%.

6 Conclusion

From what we have seen, the Moran's I statistic works very well with tracks that seem to be correlated for a long period of time. For shorter tracks, tracks with less than 4 points, it seems like the Moran's I statistic drops off and does not do a good job in determining the correlation of the track and whether or not it should be associated. By itself, the Moran's I statistic will not be sufficient to conduct automatic reconstruction but it can be a useful element in the process.

The model we developed in chapter 5 was only 86.3% accurate. This is not ideal for automatic reconstruction where we are trying to achieve greater than 99% accuracy. However, only 124 radar tracks and 4 truth objects were used. With a larger data set, it may be possible to increase the accuracy for the classification learner.

In learner this method, the idea was never that this would be that only tool needed for automatic reconstruction. There is still more research to be done in solving that problem. In this paper, we did the Moran's I for entire tracks and did not segment the track to determine if there was association for only portions of the tracks. In addition, we could do further analysis on changing up the spatial weights matrix to account for more neighbors to be analyze. We hope that the work done in this paper can contributing to the overarching goal of automatic reconstruction.

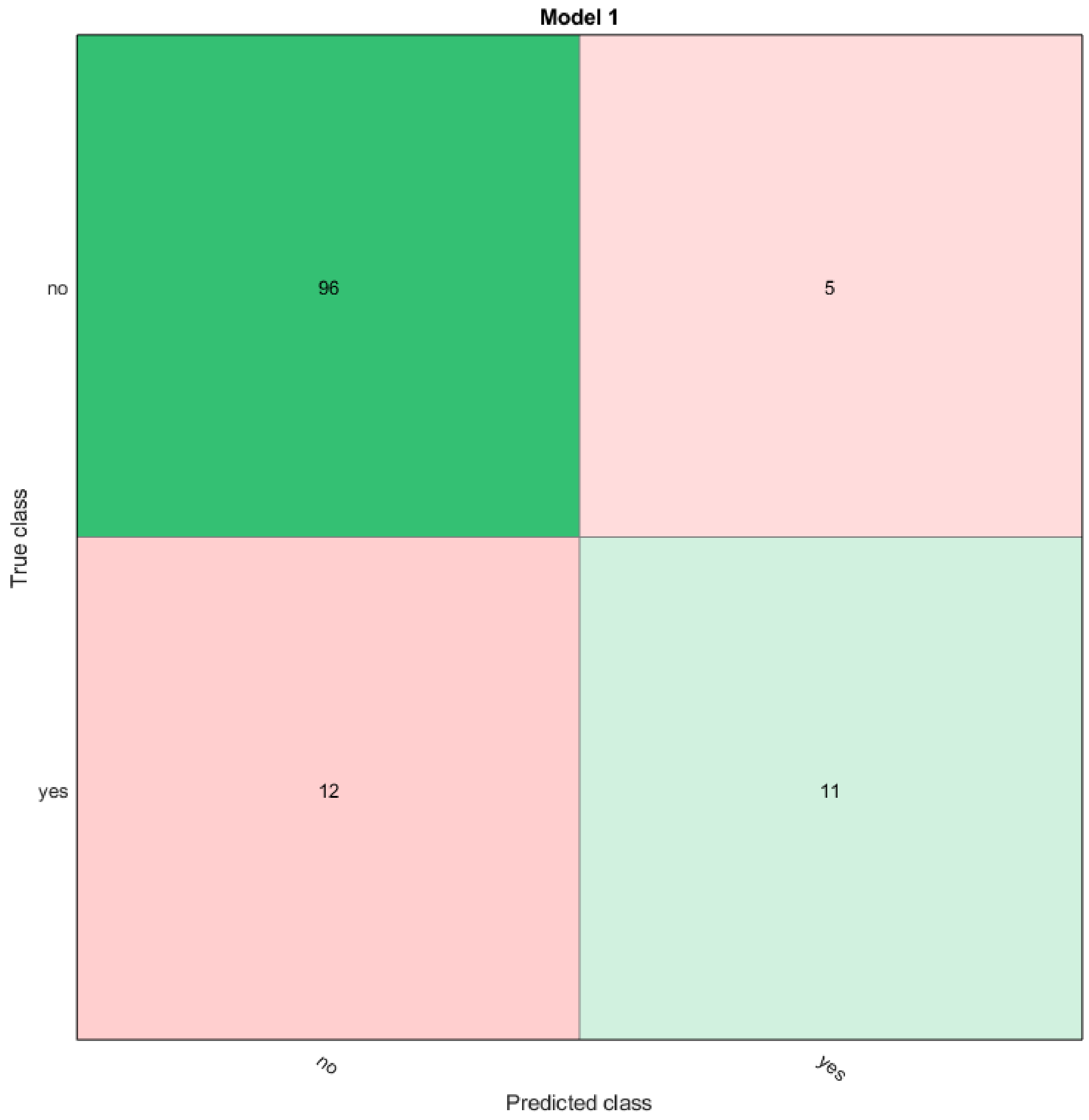


Figure 9: Confusion Matrix

References

[1] Jean Dube, Diego Legros, *Spatial Autocorrelation*, September 2014.

Appendices

A MATLAB code for figures

Listing 1: `Project Main`

```
%% Import Data

[T, T_truth, T_combined] = import_data_project ();

%% Calculations and plots

unique_object = unique(T_truth.object);
unique_ctsl = unique(T.ctsl);
T_rollup = table();

for i = 1:length(unique_object)
    for j = 1:length(unique_ctsl)

        %Finding if it is correlated
        T_ctsl = T(T.ctsl == unique_ctsl(j),:);
        unique_ctsl_object = unique(T_ctsl.object);
        if strcmp(unique_ctsl_object, unique_object(i))
            correlated = {'yes'};
        else
            correlated = {'no'};
        end

        object = unique_object(i);
        ctsl = unique_ctsl(j);

        [I_x, I_y, I_z, I_vx, I_vy, I_vz] = calculations_and_plots (T, T_truth, object, ctsl);

        T_rollup_temp = table(object, ctsl, I_x, I_y, I_z, I_vx, I_vy, I_vz, correlated);
        T_rollup = [T_rollup; T_rollup_temp];
    end
end

%% Machine Learning

```

```
ML_correlated = trainedModel.predictFcn(T_rollup);

T_rollup_ML = T_rollup;
T_rollup_ML.ML_correlated = ML_correlated;

%% View decision tree

%view(trainedModel.ClassificationTree,'mode','graph') % graphic description
```

Listing 2: Calculations and Plots

```
function [MoransI_x, MoransI_y, MoransI_z, MoransI_vx, MoransI_vy, MoransI_vz] ...
    = calculations_and_plots (T, T_truth, object, ctsl)
%% Define tracks to analyze

T_object = T_truth(strcmp(T_truth.object, object),:);
T_ctsl = T(T.ctsl == ctsl,:);

%% Ensuring unique time values

[¬, unique_time_index_object, ¬] = unique(T_object.time, 'first');
[¬, unique_time_index_ctsl, ¬] = unique(T_ctsl.time, 'first');

T_object = T_object(unique_time_index_object, :);
T_ctsl = T_ctsl(unique_time_index_ctsl, :);

%% Interpolate tracks

% object
min_time_T_object = min(T_object.time);
max_time_T_object = max(T_object.time);

% ctsl
min_time_T_ctsl = min(T_ctsl.time);
max_time_T_ctsl = max(T_ctsl.time);

% Interpolate
interp_time = max(min_time_T_object, min_time_T_ctsl): min(max_time_T_object, ...
    max_time_T_ctsl);
interp_time = interp_time';

```

```
%% x Interpolation

figure(1)
interp_x_object = interp1(T_object.time, T_object.x, interp_time);
plot(T_object.time, T_object.x, 'o', interp_time, interp_x_object, ':.')

hold on;

interp_x_ctsl = interp1(T_ctsl.time, T_ctsl.x, interp_time);

plot(T_ctsl.time, T_ctsl.x, 'o', interp_time, interp_x_ctsl, ':.')

legend('Truth', 'Interp Truth', 'Radar', 'Interp Radar')
xlabel('Time (sec)')
ylabel('x (m)')
title('X-Time Plot')


% Calculating Moran's I
MoransI_x = morans_i (interp_x_object, interp_x_ctsl);

annotation('textbox', 'String', ['I = ' char(string(MoransI_x))],...
           'Position',[0.15 0.85 0.2 0.06])

fprintf('MoransI_x = %0.5g \n', MoransI_x)

%% y Interpolation

figure(2)
interp_y_object = interp1(T_object.time, T_object.y, interp_time);
plot(T_object.time, T_object.y, 'o', interp_time, interp_y_object, ':.')

hold on;

interp_y_ctsl = interp1(T_ctsl.time, T_ctsl.y, interp_time);
plot(T_ctsl.time, T_ctsl.y, 'o', interp_time, interp_y_ctsl, ':.')

legend('Truth', 'Interp Truth', 'Radar', 'Interp Radar')
xlabel('Time (sec)')
ylabel('y (m)')
title('y-Time Plot')
```

```

% Calculating Moran's I
MoransI_y = morans_i (interp_y_object, interp_y_ctsl);

annotation('textbox', 'String', ['I = ' char(string(MoransI_y))], ...
           'Position',[0.15 0.85 0.2 0.06])

fprintf('MoransI_y = %0.5g \n', MoransI_y)

%% z Interpolation

figure(3)
interp_z_object = interp1(T_object.time, T_object.z, interp_time);
plot(T_object.time, T_object.z, 'o', interp_time, interp_z_object, ':.')

hold on;

interp_z_ctsl = interp1(T_ctsl.time, T_ctsl.z, interp_time);
plot(T_ctsl.time, T_ctsl.z, 'o', interp_time, interp_z_ctsl, ':.')

legend('Truth', 'Interp Truth', 'Radar', 'Interp Radar')
xlabel('Time (sec)')
ylabel('z (m)')
title('z-Time Plot')

% Calculating Moran's I
MoransI_z = morans_i (interp_z_object, interp_z_ctsl);

annotation('textbox', 'String', ['I = ' char(string(MoransI_z))],...
           'Position',[0.15 0.85 0.2 0.06])

fprintf('MoransI_z = %0.5g \n', MoransI_z)

%% vx Interpolation

figure(4)
interp_vx_object = interp1(T_object.time, T_object.vx, interp_time);
plot(T_object.time, T_object.vx, 'o', interp_time, interp_vx_object, ':.')

hold on;

```

```
interp_vx_ctsl = interp1(T_ctsl.time, T_ctsl.vx, interp_time);
plot(T_ctsl.time, T_ctsl.vx, 'o', interp_time, interp_vx_ctsl, ':.')

legend('Truth', 'Interp Truth', 'Radar', 'Interp Radar')
xlabel('Time (sec)')
ylabel('vx (m)')
title('vx-Time Plot')

% Calculating Moran's I
MoransI_vx = morans_i (interp_vx_object, interp_vx_ctsl);

annotation('textbox', 'String', ['I = ' char(string(MoransI_vx))],...
           'Position',[0.15 0.85 0.2 0.06])

fprintf('MoransI_vx = %0.5g \n', MoransI_vx)

%% vy Interpolation

figure(5)
interp_vy_object = interp1(T_object.time, T_object.vy, interp_time);
plot(T_object.time, T_object.vy, 'o', interp_time, interp_vy_object, ':.')

hold on;

interp_vy_ctsl = interp1(T_ctsl.time, T_ctsl.vy, interp_time);
plot(T_ctsl.time, T_ctsl.vy, 'o', interp_time, interp_vy_ctsl, ':.')

legend('Truth', 'Interp Truth', 'Radar', 'Interp Radar')
xlabel('Time (sec)')
ylabel('vy (m)')
title('vy-Time Plot')

% Calculating Moran's I
MoransI_vy = morans_i (interp_vy_object, interp_vy_ctsl);

annotation('textbox', 'String', ['I = ' char(string(MoransI_vy))],...
           'Position',[0.15 0.85 0.2 0.06])

fprintf('MoransI_vy = %0.5g \n', MoransI_vy)

%% vz Interpolation
```

```

figure(6)
interp_vz_object = interp1(T_object.time, T_object.vz, interp_time);
plot(T_object.time, T_object.vz, 'o', interp_time, interp_vz_object, ':.')

hold on;

interp_vz_ctsl = interp1(T_ctsl.time, T_ctsl.vz, interp_time);
plot(T_ctsl.time, T_ctsl.vz, 'o', interp_time, interp_vz_ctsl, ':.')

legend('Truth', 'Interp Truth', 'Radar', 'Interp Radar')
xlabel('Time (sec)')
ylabel('vz (m)')
title('vz-Time Plot')

% Calculating Moran's I
MoransI_vz = morans_i (interp_vz_object, interp_vz_ctsl);

%annotation('textbox', 'String', ['I = ' char(string(MoransI_vz))],...
           %'Position',[0.15 0.85 0.2 0.06])

fprintf('MoransI_vz = %0.5g \n', MoransI_vz)
```

Listing 3: `Moran's I`

```
function I = morans_i (yi, yj)
%% Calculate Moran's I

y = [yi;yj];                          % place truth and ctsl in one vector

w1 = zeros(length(yi), length(yi)); % quadrant 1 of weights
w2 = eye(length(yi));               % quadrant 2 of weights
w3 = eye(length(yi));               % quadrant 3 of weights
w4 = zeros(length(yi), length(yi)); % quadrant 4 of weights
w = [w1 w2; w3 w4];

N = length(y);                        %Number of samples
ybar = mean(y);                       % mean of all data points (truth and radar)

%Moran's I Statistic
I = N / sum(sum(w)) * (sum(sum(w .* ((y - ybar) * (y - ybar)'))))/sum((y-ybar).^2));
```

```

end
```