DOT/FAA/AR-07/67

Air Traffic Organization
Operations Planning
Office of Aviation Research
and Development
Washington, DC 20591

Analysis of Aircraft Touchdown Point and the Associated Uncertainty

January 2008

Final Report

U.S. Department of Transportation
Federal Aviation Administration

NOTICE

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/AR-07/67		

4. Title and Subtitle	5. Report Date
ANALYSIS OF AIRCRAFT TOUCHDOWN POINT AND THE ASSOCIATED UNCERTAINTY	January 2008
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Ming Ouyang, Ph.D[1], Larry Hackler[2], and Andrew Cheng, Ph.D[3]	

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
[1]Informatics Institute University of Medicine and Dentistry of New Jersey 675 Hoes Lane Piscataway, NJ 08854	1. Contract or Grant No. 2005-G-007
[2]Federal Aviation Administration William J. Hughes Technical Center Airport and Aircraft Safety Research and Development Division Flight Safety Branch Atlantic City International Airport, NJ 08405	
[3]Hi-Tec Systems, Inc. 500 Scarborough Drive, Suite 108 Egg Harbor Township, NJ 08234	

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
U.S. Department of Transportation Federal Aviation Administration Air Traffic Organization Operations Planning Office of Aviation Research and Development Washington, DC 20591	Final Report
	14. Sponsoring Agency Code AFS-408

15. Supplementary Notes

16. Abstract

Adequate determination of aircraft landing distance is critical to the safety of terminal area operations. In general, the aircraft landing process consists of the touchdown and rollout/turnoff. Although the current aviation system tracks a number of landing parameters for commercial operations, the operational touchdown performance is not readily measured and recorded. The research described herein collected landing traces from a flight simulator, analyzed the traces, and developed and validated computational methods that can backcalculate the touchdown points. The accuracy of the calculation was verified against high-precision Global Positioning System information. These computational methods are implemented in the MATLAB® script language, and they are very efficient. MATLAB provides interfaces to all commonly used database systems, including Microsoft® Access®. Thus, these methods can be readily incorporated in an integral system to process large amounts of operational landing data.

17. Key Words	18. Distribution Statement
Terminal area safety operations, Landing performance, Operational landing, Touchdown points, Landing trace	The document is available to the U.S. public through the National Technical Institute Service (NTIS), Springfield, Virginia 22161.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	85	

ACKNOWLEDGEMENT

The principal investigator thanks Archie Dillard of the Federal Aviation Administration (FAA) Flight Technologies and Procedures Division, Cheryl Cohenour of Cherokee CRC, and Mike Sies of the FAA Flight Operations Simulation and Analysis Branch for their support and assistance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

FAA	Federal Aviation Administration
GIS	Geographical Information System
GPS	Global Positioning System
LAHSO	Land and Hold Short Operations
LGA	LaGuardia Airport
U.S.	United States
UTM	Universal Transverse Mercator

EXECUTIVE SUMMARY

Adequate determination of aircraft landing distance is critical to the safety of terminal area operations. In general, the aircraft landing process consists of the touchdown and rollout/turnoff. Although the current aviation system tracks a number of landing parameters for commercial operations, the operational touchdown performance is not readily measured and recorded. The research described herein collected landing traces from a flight simulator, analyzed the traces, and developed and validated computational methods that can backcalculate the touchdown points. The accuracy of the calculation was verified against high-precision Global Positioning System information. These computational methods are implemented in the MATLAB® script language, and they are very efficient. MATLAB provides interfaces to all commonly used database systems, including Microsoft® Access®. Thus, these methods can be readily incorporated in an integral system to process large amounts of operational landing data.

1. INTRODUCTION.

Adequate determination of aircraft landing distance is critical to the safety of terminal area operations. In general, the aircraft landing process consists of the touchdown and rollout/turnoff. Although the current aviation system tracks a number of landing parameters for commercial operations, the operational touchdown performance is not readily measured and recorded. Federal Aviation Administration (FAA) of the United States (U.S.) is currently engaged in a research project to identify a range of Land and Hold Short Operations (LAHSO) landing distances needed by aircraft that currently comprise the US fleet [1]. The intent of this research was to increase the safety of LAHSO conducted under normal landing procedures. The primary research objective was to identify the relationship between the operational landing distances achieved in line operations with normal procedures and the minimum landing distances provided in Airplane Flight Manuals.

A consistent means to determine the aircraft operational landing distance from the available landing parameters was crucial to the success of this research project. The computational methods described in this report will significantly improve the identification of aircraft touchdown point, given that no direct measure of this information is available. It will also enable the FAA to compile operational landing distance information from recorded aircraft landing parameters and develop practical guidelines for aviation safety and efficiency.

A set of 72 simulated aircraft landing traces at the LaGuardia Airport (LGA) was collected specifically for this study using a Latin Square experiment design to configure the parameters of these 72 landings as a combination of factors that may potentially affect landing performance, such as aircraft weight, visibility, wind directions, and precipitation on the runway. One of the important features of simulated flight data is the availability of high-precision Global Positioning System (GPS) information. The GPS data served as the reference of aircraft locations so that algorithms can be developed and validated. In addition to the simulated landing traces, the Geographical Information System (GIS) of LGA [2] was also provided for reference.

These algorithms used the following strategy to identify the touchdown points. The runways and taxiways of LGA are divided into two segments: straight line segments and turning segments. Similarly, the landing traces are divided into straight line segments and turning segments. Then the segments of a landing trace are matched to the segments of the runways/taxiways, where the kinds of segments (straight line or turning) and the length of the segments are taken into consideration. The best match is selected, and the touchdown point is backcalculated from the match. Between the GPS touchdown points and backcalculated touchdown points, the largest discrepancy is 37.8 meters (m), the smallest is 0.6 m, and the average is 10.5 m.

The implemention of algorithms was in a suite of MATLAB® [3] scripts. It took less than 2 seconds to process each of these landing traces on a Pentium 3-GHz desktop personal computer. MATLAB is a commercial software platform. It has software toolboxes to interface with all major databases, such as Oracle® and Microsoft® Access®. Thus, the MATLAB scripts can readily be integrated into a software system that can process the massive amounts of landing data obtained from major U.S. airports [2].
This report is organized as follows:

- Section 2 describes the configuration of the landing airport, airport GIS data, and landing traces.
- Section 3 describes the computational methods.
- Section 4 presents the computational results.
- Section 5 describes the possible extension and future work.

2. AIRPORT CONFIGURATION AND SIMULATED LANDING TRACES.

2.1 AIRPORT CONFIGURATION.

Figure 1 shows the diagram of LGA. It has two runways intersecting at 90 degrees. The magnetic north of LGA was 13.3 degrees west of the polar north in January 1995, and it has shifted 0.1 degree west every year. The flight simulation that collected the landing traces used in this study was conducted in May 2006. Thus, the magnetic headings in the landing traces were adjusted by 14.4 degrees east to obtain the polar directions of the aircraft.

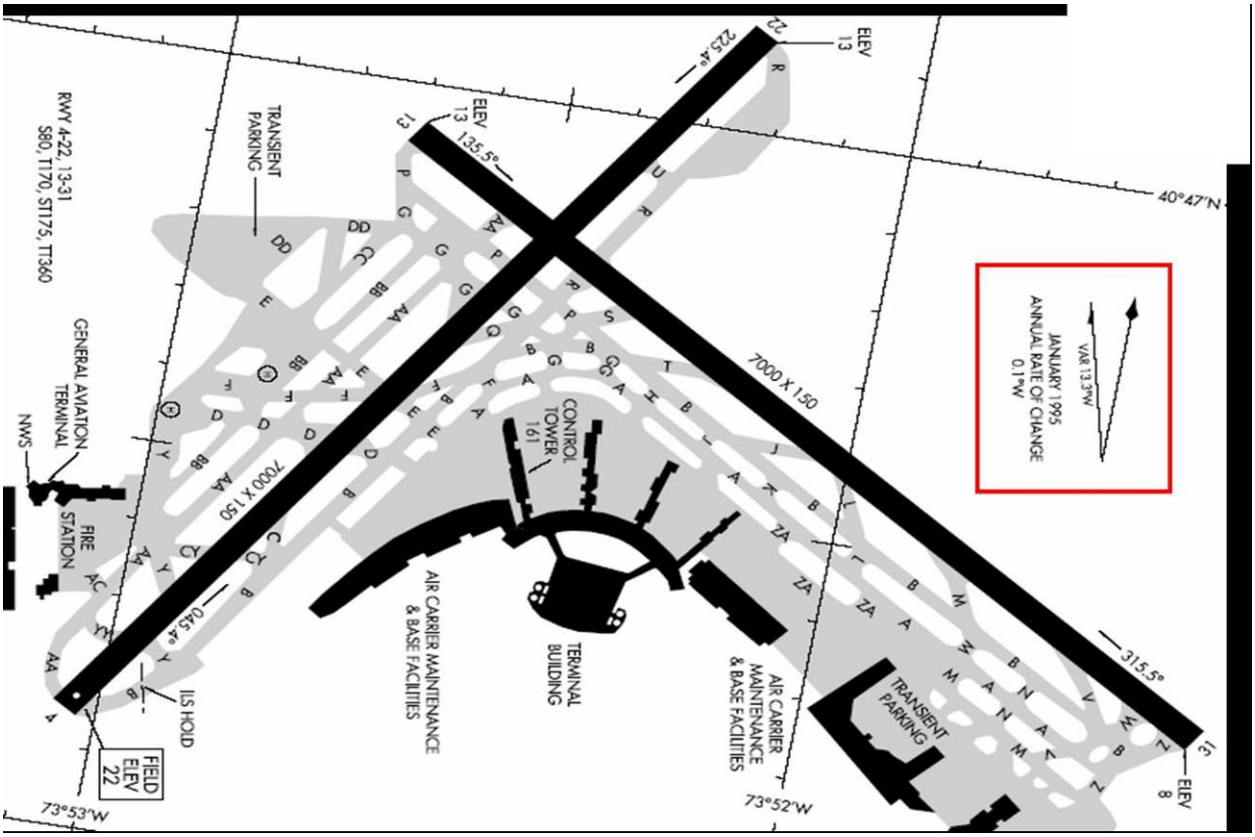

Figure 1. Diagram of LGA

The airport GIS data consist of 17 shapefiles. Shapefiles are a proprietary file format of ESRI for the storage of geographical information [4]. Six shapefiles are of particular interest:

- `LGA_Centerline`: describing the centerlines of the two runways
- `lga_runway`: describing the runways
- `lga_exitline`: describing the connections between the runways and the taxiways
- `LGA_Taxiway_Guidance_Line`: describing the centerlines of taxiways
- `lga_taxiway_segment`: describing the taxiways
- `LGA_Vertical_Polygon_Object`: describing the buildings

The geographical data in these shapefiles are represented in the latitude/longitude system, which is a spherical coordinate system involving the meridians of longitude—great circles intersecting at the poles—and the parallels of latitude. This system uses angular measurements to describe a position on the surface of the earth. However, the angular difference between two locations can not readily be converted to a distance. For example, the distance for two points that are apart by 3 degrees of longitude is not the same when these points are placed in different latitudes. The actual distance crucially depends on where the points locate from the equator. To facilitate the calculation of distances among points, the coordinate system from latitude and longitude had to be converted to the Universal Transverse Mercator (UTM) system.

The UTM projection is a transverse cylindrical projection that maps the nearly ellipsoidal earth onto the surface of a cylinder that touches the surface of the earth along a meridian of longitude, and a rectangular coordinate system, the UTM coordinates, is assigned to points on the flattened cylinder after projection. The globe is divided into zones of 6 degrees longitude with the first zone running from 180 degrees west longitude to 174 degrees west longitude. The central meridian (the meridian along which the projection cylinder contacts the surface) in each zone is assigned the arbitrary the easting coordinate of 500 kilometers, and all points within the zone are assigned coordinates based on their distance from the equator (northing) and from the hypothetical 0 point of the easting coordinate. Thus, at the equator and at the central meridian, the coordinate is (500.0 km, 0 km). Since the zones are less than 1000 kilometers wide, there is no point which is actually given the coordinate (0, 0), and all UTM coordinates are positive. Zones are also divided into 8-degree bands designated by a letter, starting at A at the South Pole and skipping the letters I and O. LGA locates in UTM zone 18N. A number of software packages can easily convert latitude/longitude coordinates to the UTM coordinates. Since the UTM coordinates are a base-ten metric coordinate system, it is a simple task to calculate distances among points. Figure 2 shows the (NAD 1983 UTM Zone 18N) projected map of LGA, showing the six aforementioned shapefiles.

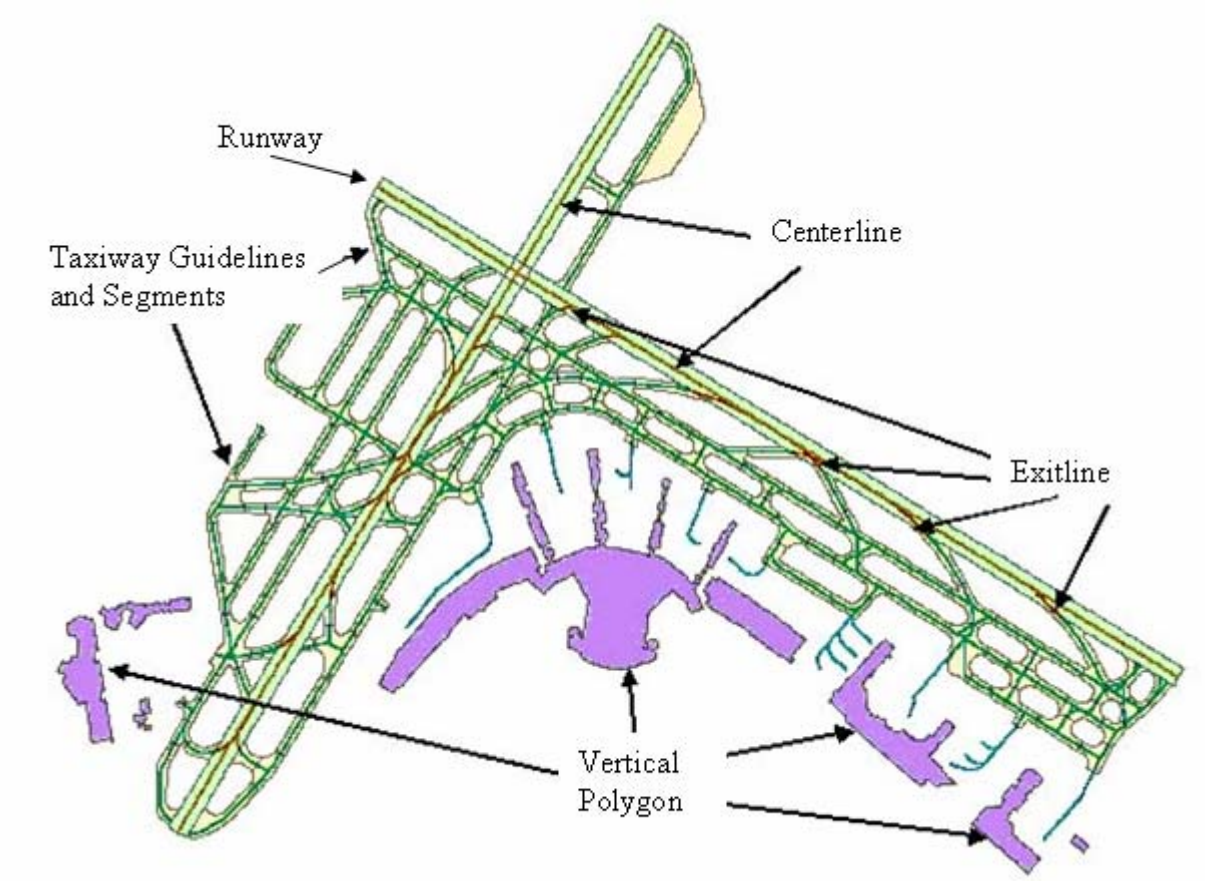

Figure 2. A GIS Diagram of LGA

2.2 FLIGHT SIMULATION.

The flight simulation was conducted at the FAA Flight Operations Simulation and Analysis Branch, Oklahoma City, OK. The study collected landing data from LGA via three different runway directions: RWY31, RWY13, and RWY22. The landing simulation started in the approach phase at approximately 3 miles from the runway threshold. After touchdown, the simulation continued through turning off and taxiing via taxiways A and B until stopping at the transient parking. The objective of this study was to collect aircraft landing parameters that represent routine operations from the final approach to stopping at the gate. Because the collected data would be used to develop and validate computational methods, specific variations were introduced into the following factors that may potentially affect landing performance:

- Runways: 13, 22, 31
- Wind: calm, headwind (10 knots), crosswind (10 knots), and tailwind (10 knots)
- Precipitation on runway: dry or wet
- Visibility: clear, moderate low
- Aircraft gross weight: 130,000 or 144,000 tons

In total, there were 72 different scenarios arranged in a Latin Square experiment design. The complete list of scenarios is shown in table 1. The purpose of the Latin Square design was to ensure that the sequential order of scenarios in the simulations did not consistently favor one particular factor/level in such a way that the obtained results would be misleading.

Table 1. The Latin Square Design of the 72 Scenarios for Flight Simulation

Scenario #	Wind	Precipitation	Visibility	Runway	Holdshort	Gross Wt.	Pilot ID
1	Calm	Dry	Clear	31	N	144,000	1
2	Calm	Wet	M. Low	31	N	144,000	1
3	Head	Wet	M. Low	31	N	144,000	1
4	Tail	Wet	M. Low	31	Y	144,000	1
5	Calm	Dry	M. Low	31	Y	130,000	1
6	Head	Dry	M. Low	31	N	130,000	1
7	Tail	Dry	M. Low	31	N	130,000	1
8	Cross	Wet	Clear	31	N	130,000	1
9	Calm	Dry	Clear	13	N	144,000	1
10	Calm	Wet	M. Low	13	N	144,000	1
11	Head	Wet	Clear	13	Y	144,000	1
12	Tail	Wet	M. Low	13	N	144,000	1
13	Calm	Dry	M. Low	13	Y	130,000	1
14	Head	Dry	Clear	13	Y	130,000	1
15	Tail	Dry	M. Low	13	N	130,000	1
16	Cross	Wet	Clear	13	N	130,000	1
17	Calm	Dry	Clear	22	N	144,000	1
18	Head	Dry	Clear	22	N	144,000	1
19	Head	Wet	M. Low	22	Y	144,000	1
20	Tail	Wet	M. Low	22	N	144,000	1
21	Calm	Dry	M. Low	22	Y	130,000	1
22	Head	Wet	Clear	22	N	130,000	1
23	Tail	Dry	M. Low	22	N	130,000	1
24	Cross	Wet	Clear	22	N	130,000	1
25	Calm	Wet	Clear	31	N	144,000	2
26	Head	Wet	Clear	31	N	144,000	2
27	Tail	Wet	Clear	31	N	144,000	2
28	Cross	Wet	M. Low	31	N	144,000	2
29	Calm	Dry	Clear	31	Y	130,000	2
30	Head	Dry	Clear	31	N	130,000	2
31	Head	Wet	M. Low	31	Y	130,000	2
32	Cross	Dry	Clear	31	N	130,000	2
33	Calm	Wet	Clear	13	N	144,000	2
34	Head	Dry	M. Low	13	N	144,000	2
35	Tail	Wet	Clear	13	N	144,000	2
36	Cross	Wet	M. Low	13	N	144,000	2
37	Calm	Dry	Clear	13	Y	130,000	2
38	Calm	Wet	M. Low	13	Y	130,000	2
39	Head	Wet	M. Low	13	N	130,000	2
40	Cross	Dry	Clear	13	N	130,000	2
41	Calm	Wet	Clear	22	N	144,000	2
42	Head	Wet	Clear	22	Y	144,000	2
43	Tail	Dry	M. Low	22	Y	144,000	2
44	Cross	Wet	Clear	22	Y	144,000	2
45	Calm	Dry	Clear	22	Y	130,000	2
46	Head	Dry	M. Low	22	N	130,000	2
47	Tail	Dry	Clear	22	N	130,000	2
48	Cross	Dry	Clear	22	N	130,000	2
49	Calm	Dry	M. Low	31	N	144,000	3
50	Head	Dry	Clear	31	Y	144,000	3
51	Tail	Dry	Clear	31	N	144,000	3
52	Cross	Dry	M. Low	31	N	144,000	3
53	Calm	Wet	Clear	31	Y	130,000	3
54	Head	Wet	Clear	31	Y	130,000	3
55	Tail	Wet	M. Low	31	N	130,000	3
56	Cross	Wet	M. Low	31	Y	130,000	3
57	Calm	Dry	M. Low	13	N	144,000	3
58	Head	Dry	Clear	13	N	144,000	3
59	Tail	Dry	Clear	13	N	144,000	3
60	Cross	Dry	M. Low	13	N	144,000	3
61	Calm	Wet	Clear	13	Y	130,000	3

Table 1. The Latin Square Design of the 72 Scenarios for Flight Simulation (Continued)

Scenario #	Wind	Precipitation	Visibility	Runway	Holdshort	Gross Wt.	Pilot ID
62	Head	Wet	Clear	13	N	130,000	3
63	Tail	Wet	Clear	13	Y	130,000	3
64	Cross	Wet	M. Low	13	Y	130,000	3
65	Calm	Dry	M. Low	22	N	144,000	3
66	Head	Dry	M. Low	22	Y	144,000	3
67	Tail	Dry	Clear	22	Y	144,000	3
68	Cross	Dry	M. Low	22	N	144,000	3
69	Calm	Wet	M. Low	22	N	130,000	3
70	Head	Wet	M. Low	22	N	130,000	3
71	Tail	Wet	Clear	22	N	130,000	3
72	Cross	Wet	M. Low	22	N	130,000	3

2.3 SIMULATED LANDING TRACES.

The flight simulator (figure 3) recorded a wealth of flight information including this project, ground speed (knots), main gear squat (0 = in air, -1 = on ground), aircraft magnetic heading (degrees), aircraft latitude (degrees), and aircraft longitude (degrees).

Figure 3. Flight Simulator

For data analysis, there are two major differences between operational and simulated landing traces. The first is the frequency that the data are recorded: 1 Hz for operational data and 5 Hz for simulation data. The second is the precision of the recorded data. For example, in the operational data, the latitude and longitude have 3 digits after the decimal (in degrees), or a resolution of 275 feet, whereas in the simulation data, there are 13 digits after the decimal, or a resolution of 1.6 feet. In other words, the data from the flight simulator painted a very detailed picture of flight information. Therefore, when the data analysis was performed (described in the next section), four out of every five records were discarded to reduce the data sampling rate to 1 Hz. The numbers were rounded up to match the corresponding precision in the operational data. Of the 72 recorded data files, one was corrupted, leaving 71 usable simulated landing traces.

3. ANALYSIS OF SIMULATED LANDING TRACES.

In a landing trace, the touchdown point in time was identified as the first record where the squat switch was closed. The data prior to this point were not used for the rest of the study. Because the data sampling rate was 5 Hz for the simulated data and 1 Hz for the operational data, every fifth record was kept from the touchdown point to the parking position to match the coarse sampling rate in operational data.

3.1 THE REFERENCE: HIGH-PRECISION GPS LATITUDE/LONGITUDE.

The GPS latitude/longitude data from the flight simulator have a resolution of 1.6 feet, and they were used as the reference of the aircraft location. The conversion from latitude/longitude to UTM coordinates is straightforward algebra and trigonometry, and it is built into many software tools. The “utmtool” was used for the conversion. The utmtool was developed and maintained by the Community Modeling & Analysis System Center, University of North Carolina at Chapel Hill. As a measure of quality check, the 71 traces of UTM coordinates (from touchdown to parking) were plotted with the UTM-projected LGA map, as shown in figure 4. It can be readily seen that these GPS traces are of sufficiently high precision and can serve as the references of aircraft locations.

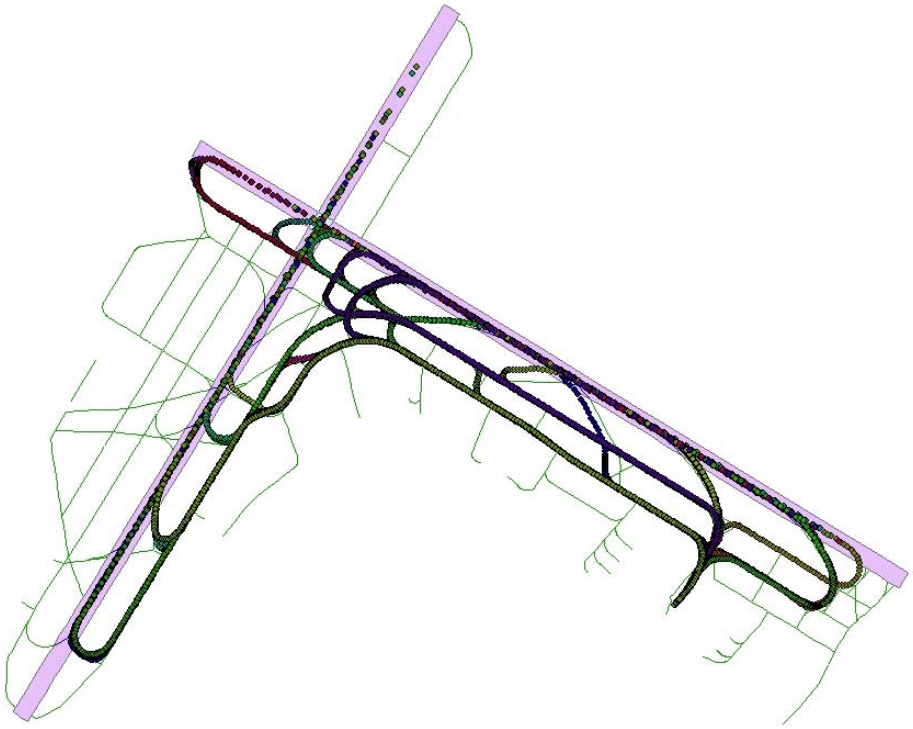

Figure 4. All 71 Simulated Landing Traces in LGA

3.2 CONVERSION FROM SPEED/HEADING TO MOVEMENT.

Although the flight simulator recorded a wealth of information about the flight, only the ground speed (knots) and the magnetic heading (degrees) in the reconstruction of the path of the aircraft and the determination of the touchdown point will be used. As mentioned earlier, retaining every fifth of the records from the simulated data was necessary in keeping with the coarse sampling rate of 1 Hz in operational data. Furthermore, the numbers of ground speed and magnetic heading were rounded up to match the precision that is commonly available in operational data. The next task was to infer the whereabouts of the aircraft from the (coarse and imprecise) speed/heading data.

When the aircraft was accelerating or decelerating, the ground speeds (V) between two consecutive records showed considerable difference; similarly, when the aircraft was making turns, consecutive magnetic headings (H) exhibited rapid changes. The comparation of two methods (linear fitting and cubic spline fitting) that transform speed/heading into movement was executed while dealing with the rapid changes.

With linear fitting, the average of two consecutive ground speeds (knots) was used to obtain the distance of the movement, which was then multiplied by 0.51444444 to convert to meters per second. The average of two consecutive magnetic headings was used for the direction of the movement. Let (X_0, Y_0) be the coordinates of the touchdown point; its coordinates were arbitrarily assigned to (0, 0) for illustration purposes, while its true values are to be determined.

The coordinates for the subsequent aircraft location at time t (X_t, Y_t) can then be computed iteratively as follows:

$$X_t = X_{t-1} + \frac{V_t + V_{t-1}}{2} \cdot \sin\left(\frac{\pi}{180} \cdot \frac{H_t + H_{t-1}}{2}\right)$$

$$Y_t = Y_{t-1} + \frac{V_t + V_{t-1}}{2} \cdot \cos\left(\frac{\pi}{180} \cdot \frac{H_t + H_{t-1}}{2}\right)$$

for t = 1, 2, …, etc.

Note that aircraft speeds and traveling distances are calculated in meters, for conformation to the decimal system in UTM projection. These speeds and distances can be converted to units in feet by using the formula: 1 meter = 3.2808399 feet.

With cubic spline fitting, the movement from time s to time $s+1$ is calculated from the speed/heading of four time points, $s-1$, s, $s+1$, and $s+2$, as follows. First, the instantaneous speed/heading at each time point directly converts the movements in the x and y directions:

$$X_t = V_t \cdot \sin\left(\frac{\pi}{180} \cdot H_t\right)$$

$$Y_t = V_t \cdot \cos\left(\frac{\pi}{180} \cdot H_t\right)$$

for $t = s-1$, s, $s+1$, and $s+2$. Then, a cubic polynomial is fitted to the four (instantaneous) movements in the x direction:

$$X_s(t) = a_s + b_s t + c_s t^2 + d_s t^3$$

Then, the movement of the aircraft in the x direction from time s to time $s+1$ is the area under the curve $X_s(t)$, which is calculated by numerical integration. The movement in the y direction is similarly calculated.

It was easy to determine which of linear fitting and cubic spline fitting produces better results by comparing them to the GPS reference. Since the difference in second-by-second movements would be very small, the total differences accumulated in the final parking positions were compared. Figure 5 shows the histograms of the distances between the reconstructed parking positions and the GPS parking positions. It is clear that the cubic spline fitting is much better than the linear fitting. Specifically, the cubic spline fitting is the better method with a p-value of 10^{-15} as determined by the paired t-test.

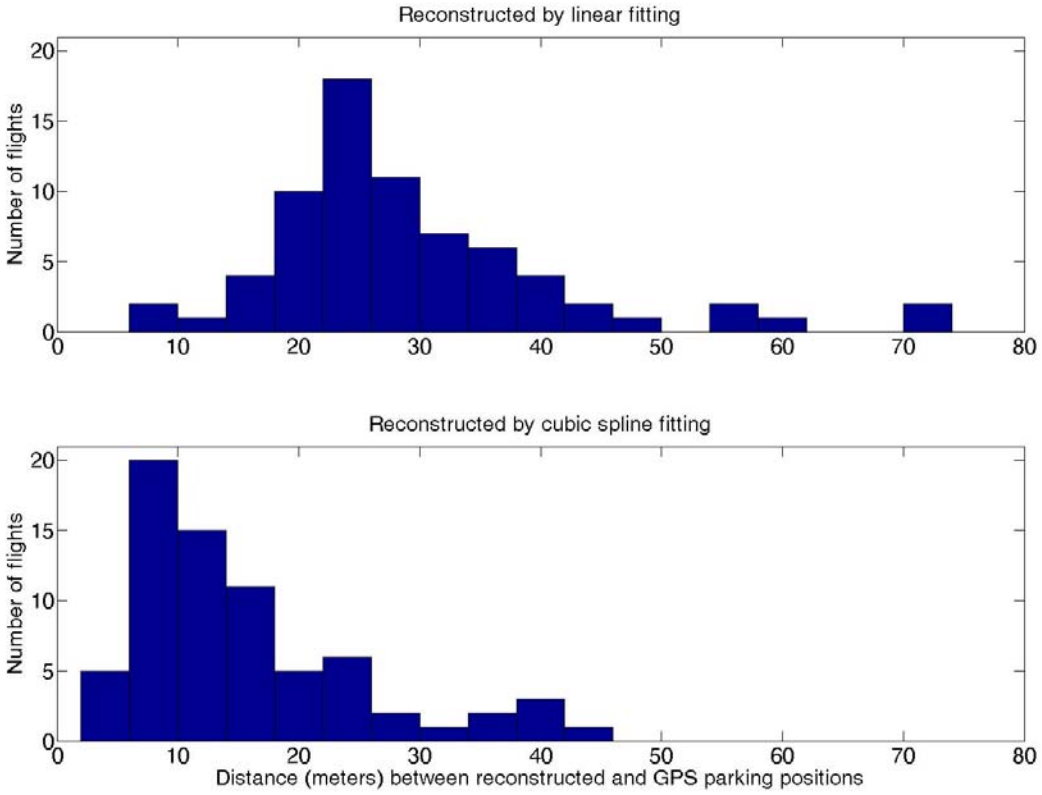

Figure 5. Histograms of Distances Between Reconstructed and GPS Parking Positions

Figure 6 shows the geographical representation of a landing trace. (The (X, Y) coordinates are in meters.) When the aircraft was parallel to the runways, the trace was plotted with black dots, and when the aircraft was making turns, it was plotted with red dots.

The (X, Y) coordinates are in meters. The touchdown point is assigned (0, 0) for the purpose of illustration; its true coordinates are to be determined. When the aircraft was parallel to the runways, the trace was plotted with black dots, and when the aircraft was making turns, it was plotted with red dots.

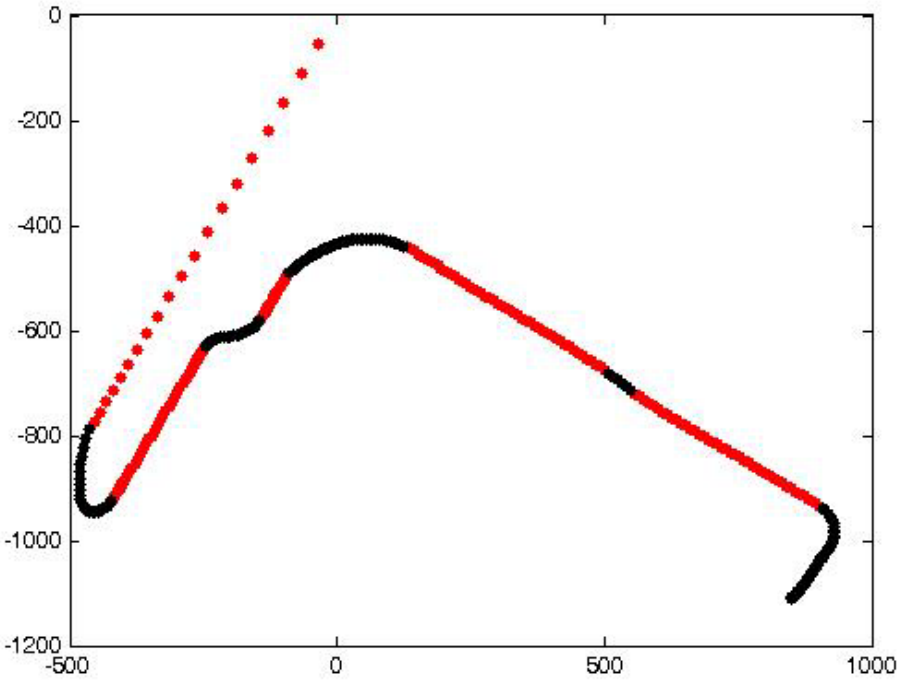

Figure 6. The Geographical Representation of a Landing Trace

3.3 OVERLAY OF TRACES TO AIRPORT CONFIGURATION.

Having converted a landing trace to a geographical representation, the next task was to overlay the representation to the airport configuration. First, prepare the airport GIS map as follows (figure 7):

1. On the airport GIS map, put red marks at points on the runways and the taxiways where aircraft may make turns.

2. The red marks are organized in an acyclic-directed network, and each red mark is a node in the network.

3. The source of the network is the runway.

    a. The children of the source are the five exit points.

    b. The grandchildren of the source are the red marks on the first taxiway parallel to the runway.

    c. The other red marks are similarly organized, and they all converge to the drain of the network.

4. The drain of the network is the terminal area.

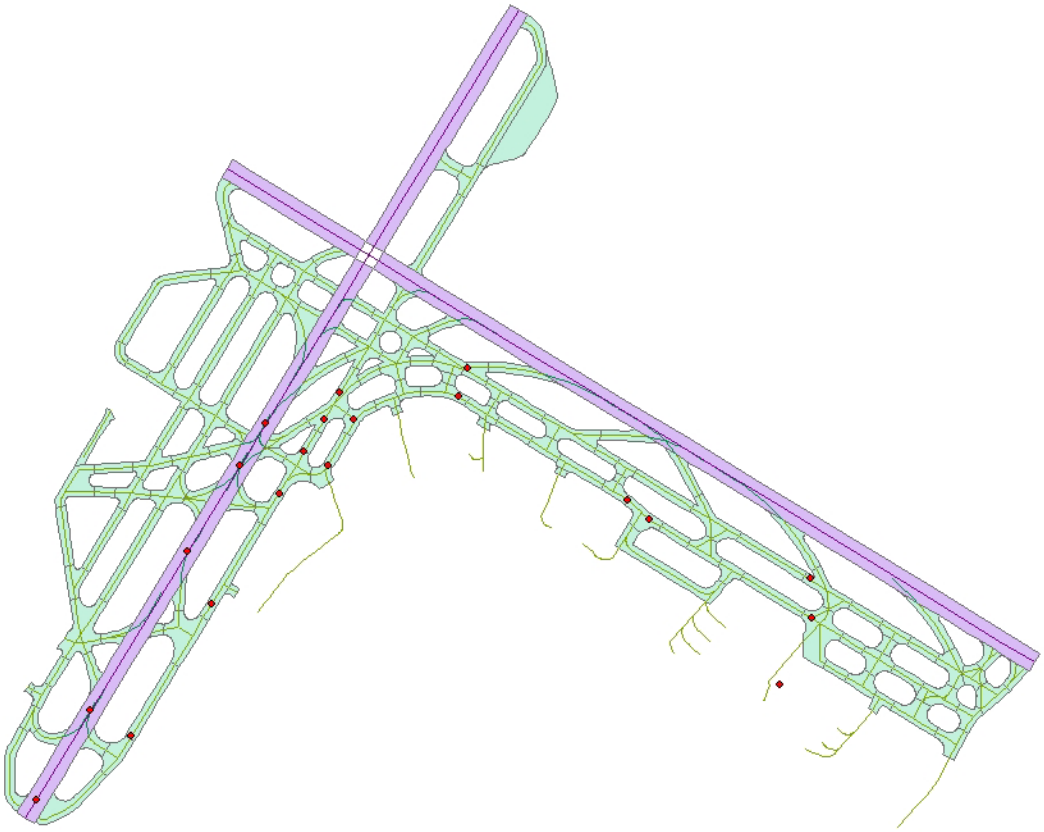

Figure 7. The Red Marks on the Runways and Taxiways are Points Where Aircraft May Make Turns

As shown in figure 6, a landing trace is partitioned into two kinds of movements: straight line movements and turning movements; a movement of one kind is always followed by the other kind. Furthermore, the red marks in figure 7 are points where aircraft may make turns. Thus, each segment of a landing trace should be anchored at one of the red marks. Specifically, the method to overlay a landing trace to the airport configuration consists of the following steps:

1. Initialize the total deviation to zero.

2. From the source of the acyclic-directed network (the runway), take the first segment of a landing trace and anchor it at one of the children of the source (the five red marks on the runway).

3. Take the second segment and anchor it at one of the grandchildren of the source (the red marks on the first taxiway parallel to the runway).

   With the correct overlay of a landing trace on the airport configuration in an ideal situation, when one end of the segment is anchored at a red mark, the other end of the segment should coincide with the next red mark. In reality, the distance between the other end of the segment and the next red mark would tell how well the segment fits this part of the airport configuration.

4. Add the distance to the total deviation.

5. Repeat steps 3 and 4 for all subsequent segments of the landing trace and for all red marks of the acyclic-directed network. Accumulate the distance between the other end of the segment and the next red mark in the total deviation.

6. The best overlay is the one with the smallest total deviation.

After the best overlay is determined, it is easy to backcalculate the coordinates of the touchdown point and compare them to the coordinates of the GPS touchdown point. The accuracy of the calculation is of the most interest. Figure 8 is the histogram of the distances between the calculated touchdown points and the GPS touchdown points. The largest distance is 37.8 meters, the smallest is 0.6 meter, and the average is 10.5 meters. As a reference, the typical touchdown speed is 140 knots, or 72 meters per second.

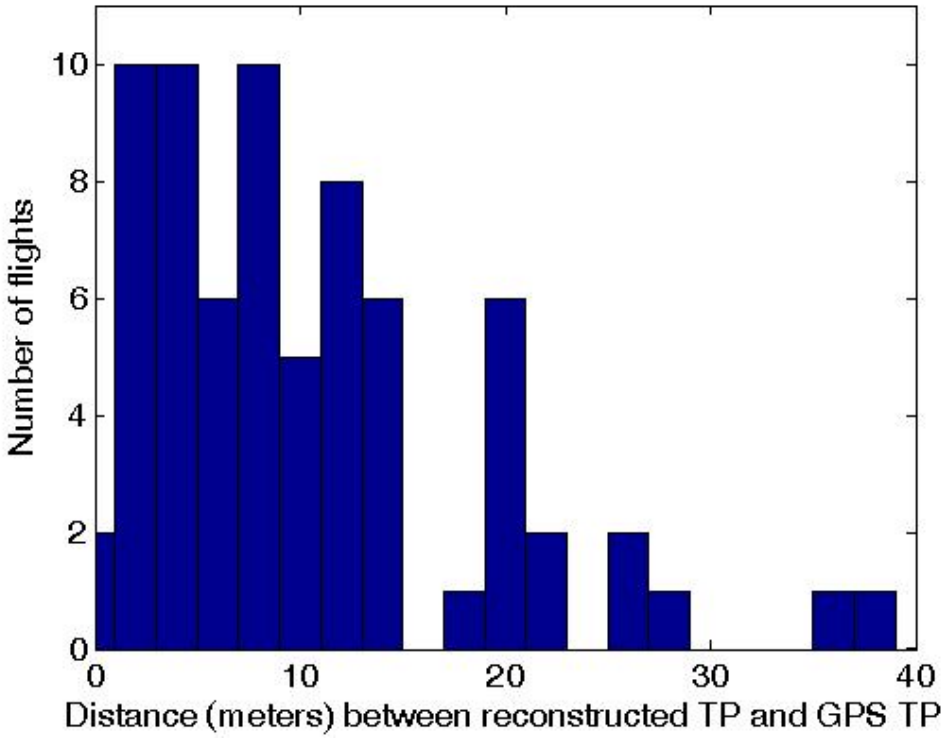

Figure 8. Histogram of Distances Between Calculated Touchdown Points and GPS Touchdown Points

4. DISCUSSION.

In recorded data from operational landing, the status of the squat switch is updated once per second. Thus, there is the possibility of up to 1 second of delay from the touchdown point in time to the first record in the data showing that the switch is closed. At the typical touchdown

speed of 140 knots, the aircraft may have traveled up to 72 meters during this period of time. This distance is the minimum uncertainty in calculated touchdown points.

In this study, the high-precision GPS latitude/longitude information in simulated flight data was used as the reference. Thus, it is able to develop and validate novel computational methods. The initial results were very encouraging. The average of the distances between the calculated touchdown points and the GPS touchdown points was 10.5 meters, and the largest such distance was 37.8 meters.

The computational methods are implemented in MATLAB scripts, which are very efficient. It takes less than 2 seconds to process one landing trace. MATLAB provides direct interface with Microsoft Access. Thus, it can readily help to process large amounts of operational data when they become available.

5. REFERENCES.

1. Terminal Area Safety Program Fact Sheet, FAA William J. Hughes Technical Center, AJP-6350.

2. FAA Airport GIS: ftp://aosftp.jccbi.gov/230/public/SF21/.

3. MathWorks: http://www.mathworks.com/.

4. ESRI shapefile technical description: http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

APPENDIX A—MATLAB® SCRIPTS IMPLEMENTING THE COMPUTATIONAL METHODS

A.1. THE MAIN SCRIPT OF THE COMPUTATIONAL METHODS.

```
function [deviation,estTP,trueTP] = analyzeTrace(f,option)

files = [110011 110041 110071 110121 110171 110201 110241
110281 110331 110371 ...
    110401 111021 111051 111101 111151 111181 111221 111261
111311 111341 ...
    111391 111421 111471 120032 120061 120111 120161 120191
120231 120271 ...
    120321 120351 120391 120421 120471 121011 121041 121081
121091 121131 ...
    121171 121241 121291 121331 121381 121411 121451 130021
130051 130091 ...
    130141 130181 130211 130251 130301 130341 130381 130411
130461 131031 ...
    131071 131151 131162 131191 131231 131281 131321 131361
131401 131431 ...
    131481];

if (nargin==1)
    verbosity = 0;
else
    switch option
        case {'off','none','noGraphics'}
            verbosity = 0;
        case 'someGraphics'
            verbosity = 1;
        case {'Graphics', 'graphics'}
            verbosity = 2;
        otherwise
            verbosity = 1;
    end
end

NWthreshold = [594628.3 4515195.9]; SWthreshold = [594174.7
4513731.8];
SEthreshold = [596450.2 4514082.9]; NEthreshold = [595286.5
4515553.0];
NWdir = acos((SEthreshold(1)-NWthreshold(1)) / ...
    sqrt(sum((NWthreshold-SEthreshold).^2))) * 180/pi + 270;
SEdir = NWdir - 180;
```

```
NEdir = 90 - acos((NEthreshold(1)-SWthreshold(1)) / ...
    sqrt(sum((NEthreshold-SWthreshold).^2))) * 180/pi;
SWdir = NEdir + 180;
MHcorrection = 14.2;
data = load(strcat('../Data/',int2str(files(f)),'.txt'));
heading = data(1,2)-MHcorrection;
if (heading<0) heading = heading+360; end

if (abs(heading-NWdir)<5.5)
    [deviation,estTP,trueTP] = predictNW(data,verbosity);
else if (abs(heading-SEdir)<5.5)
        [deviation,estTP,trueTP] =
predictSE(data,verbosity);
    else if (abs(heading-SWdir)<5.5)
            [deviation,estTP,trueTP] =
predictSW(data,verbosity);
        else
            fprintf(1,'wrong direction\n');
        end
    end
end
```

A.2. THE FUNCTION THAT USES CUBIC SPLINE TO FIT THE MAGNETIC HEADING AND GROUND SPEED.

```
function [x,y] = cubicUTM(speed,MH)

knotToMeterPerSec = 0.51444444;
numPoint = length(speed);
X = zeros(numPoint,1);
Y = zeros(numPoint,1);
disX = speed .* knotToMeterPerSec .* sin(pi*MH/180);
disY = speed .* knotToMeterPerSec .* cos(pi*MH/180);

i = 1;
coef = polyfit([0 1 2]',disX(i:(i+2)),2);
f = @(t)(coef(1)*t.^2+coef(2)*t+coef(3));
X(i) = quad(f,0,1);
coef = polyfit([0 1 2]',disY(i:(i+2)),2);
f = @(t)(coef(1)*t.^2+coef(2)*t+coef(3));
Y(i) = quad(f,0,1);

for i=2:(numPoint-2)
    coef = polyfit([-1 0 1 2]',disX((i-1):(i+2)),3);
    f = @(t)(coef(1)*t.^3+coef(2)*t.^2+coef(3)*t+coef(4));
```

```
    X(i) = quad(f,0,1);
    coef = polyfit([-1 0 1 2]',disY((i-1):(i+2)),3);
    f = @(t)(coef(1)*t.^3+coef(2)*t.^2+coef(3)*t+coef(4));
    Y(i) = quad(f,0,1);
end

i = numPoint-1;
coef = polyfit([-1 0 1]',disX((i-1):(i+1)),2);
f = @(t)(coef(1)*t.^2+coef(2)*t+coef(3));
X(i) = quad(f,0,1);
coef = polyfit([-1 0 1]',disY((i-1):(i+1)),2);
f = @(t)(coef(1)*t.^2+coef(2)*t+coef(3));
Y(i) = quad(f,0,1);

X(numPoint) = disX(numPoint);
Y(numPoint) = disY(numPoint);

x = zeros(numPoint,1);
y = zeros(numPoint,1);

for i=2:numPoint
    x(i) = x(i-1) + X(i-1);
    y(i) = y(i-1) + Y(i-1);
end
```

A.3. THE FUNCTION THAT CLASSIFIES A LANDING TRACES INTO SEGMENTS OF STRAIGHT LINE OR TURNING MOVEMENTS.

```
function [numSeg,seg] = segment(dir)

seg = zeros(10,2);
numSeg = 0;
i = 0;
want = dir(1);
while ((i+1)<=length(dir))
    i = i+1;
    numSeg = numSeg+1;
    seg(numSeg,1) = i;
    while ((i+1)<=length(dir) & (dir(i+1)==want))
        i = i+1;
    end
    seg(numSeg,2) = i;
    want = abs(want-1);
end
```

A.4. THE FUNCTION THAT PROCESSES LANDING TRACES ON RUNWAY 31.

```
function [deviation,estTP,trueTP] =
predictNW(data,verbosity)

node = [
    595647 4514573; ... % 1
    595461 4514687; ... % 2
    595193 4514850; ... % 3
    595103 4514905; ... % 4
    594997 4514970; ... % 5 runway intersect, first
    594662 4515175; ... % 6 end of runway
    594646 4515059; ... % 7
    594828 4514949; ... % 8
    594937 4514883; ... % 9
    595014 4514836; ... %10
    595079 4514796; ... %11
    595172 4514739; ... %12
    595213 4514714; ... %13
    595460 4514563; ... %14
    595944 4514272; ... %15
    595946 4514181; ... %16
    595582 4514403; ... %17
    595533 4514447; ... %18
    595372 4514545; ... %19
    595153 4514678; ... %20
    595341 4514636; ... %21
    594948 4515000; ... %22 runway intersect, second
    595875 4514033 % parking position
    ];
NWthreshold = [594628.3 4515195.9]; SWthreshold = [594174.7
4513731.8];
SEthreshold = [596450.2 4514082.9]; NEthreshold = [595286.5
4515553.0];
NWdir = acos((SEthreshold(1)-NWthreshold(1)) / ...
    sqrt(sum((NWthreshold-SEthreshold).^2))) * 180/pi + 270;
SEdir = NWdir - 180;
NEdir = 90 - acos((NEthreshold(1)-SWthreshold(1)) / ...
    sqrt(sum((NEthreshold-SWthreshold).^2))) * 180/pi;
SWdir = NEdir + 180;

MHcorrection = 14.2;
wrong = 9999; deepest = 10;
```

```
trueTP = data(1,5:6);
speed = round(data(:,1));
MH = round((data(:,2)-MHcorrection) * 100) / 100;
MH(MH<0) = MH(MH<0)+360;
[X,Y] = cubicUTM(speed,MH);
nw_se = (abs(MH-NWdir)<5.5) | (abs(MH-SEdir)<5.5);
while (1)
    [numSeg,seg] = segment(nw_se);
    again = 0;
    for i=1:numSeg
        if ((seg(i,2)-seg(i,1)+1)<=3)
            nw_se(seg(i,1):seg(i,2)) = abs(nw_se(seg(i,1))-
1);
            again = 1;
        end
    end
    if (again==0) break; end
end

stackX{1} = linspace(NWthreshold(1),SEthreshold(1),100);
stackY{1} = linspace(NWthreshold(2),SEthreshold(2),100);
stackG{1} = '-b';
stackX{2} = linspace(SWthreshold(1),NEthreshold(1),100);
stackY{2} = linspace(SWthreshold(2),NEthreshold(2),100);
stackG{2} = '-b';
stackX{3} = node(:,1); stackY{3} = node(:,2); stackG{3} =
'*b'; stackSize = 3;
if (verbosity==2) plotGraphics; end
stackX{4} = data(:,5); stackY{4} = data(:,6); stackG{4} =
'.k'; stackSize = 4;
if (verbosity>0) plotGraphics; end
for i=1:numSeg
    stackX{i+3} = X(seg(i,1):seg(i,2))+trueTP(1);
    stackY{i+3} = Y(seg(i,1):seg(i,2))+trueTP(2);
    if (mod(i,2)==1) stackG{i+3} = '.r'; else stackG{i+3} =
'.k'; end
end
stackSize = numSeg+3;
if (verbosity>0) plotGraphics; end
stackSize = 3;

% search
[deviation,path] = runway;
% search
```

```
for i=1:(numSeg-1)
    stackX{i+3}   =   X(seg(i,1):seg(i,2))+node(path(1,i),1)-
X(seg(i,2));
    stackY{i+3}   =   Y(seg(i,1):seg(i,2))+node(path(1,i),2)-
Y(seg(i,2));
    stackG{i+3} = '.k';
end
%i = numSeg;
%stackX{i+3} = X(seg(i,1):seg(i,2))+node(end,1)-X(seg(i,2));
%stackY{i+3} = Y(seg(i,1):seg(i,2))+node(end,2)-Y(seg(i,2));
%stackG{i+3} = '.k';
%stackSize = numSeg+3;
stackSize = numSeg+3-1;
for i=1:(numSeg-1)
    stackX{stackSize+i} = node(path(1,i),1);
    stackY{stackSize+i} = node(path(1,i),2);
    stackG{stackSize+i} = '*r';
end
%i = numSeg;
%stackX{stackSize+i} = node(end,1);
%stackY{stackSize+i} = node(end,2);
%stackG{stackSize+i} = '*r';
%stackSize = stackSize+numSeg;
stackSize = stackSize+numSeg-1;
if (verbosity==2) plotGraphics; end

estTP   =   fminsearch(@totalDev,node(path(1),:)-[X(seg(1,2))
Y(seg(1,2))], ...
    [],node(path(1:numSeg-1),:),[X(seg(1:numSeg-1,2))
Y(seg(1:numSeg-1,2))]);

stackX{4} = X+estTP(1);
stackY{4} = Y+estTP(2);
stackG{4} = '.k';
stackSize = 4;
for i=1:(numSeg-1)
    stackX{stackSize+i} = node(path(1,i),1);
    stackY{stackSize+i} = node(path(1,i),2);
    stackG{stackSize+i} = '*r';
end
%i = numSeg;
%stackX{stackSize+i} = node(end,1);
%stackY{stackSize+i} = node(end,2);
%stackG{stackSize+i} = '*r';
%stackSize = stackSize+numSeg;
```

```
stackSize = stackSize+numSeg-1;
if (verbosity==2) plotGraphics; end

coef = polyfit([NWthreshold(1)
SEthreshold(1)]',[NWthreshold(2) SEthreshold(2)]',1);
correction = (estTP(2)-coef(1)*estTP(1)-
coef(2))/(2*coef(1));
estTP(1) = estTP(1)+correction;
estTP(2) = estTP(2)-coef(1)*correction;

stackX{4} = X+estTP(1);
stackY{4} = Y+estTP(2);
stackG{4} = '.k';
stackSize = 4;
for i=1:(numSeg-1)
    stackX{stackSize+i} = node(path(1,i),1);
    stackY{stackSize+i} = node(path(1,i),2);
    stackG{stackSize+i} = '*r';
end
%i = numSeg;
%stackX{stackSize+i} = node(end,1);
%stackY{stackSize+i} = node(end,2);
%stackG{stackSize+i} = '*r';
%stackSize = stackSize+numSeg;
stackSize = stackSize+numSeg-1;
if (verbosity==2) plotGraphics; end

stackX{5} = trueTP(1);
stackY{5} = trueTP(2);
stackG{5} = '*r';
stackSize = 5;
if (verbosity>0) plotGraphics; end

fprintf(1,'GPS:(%d,%d) ',round(trueTP(1)),round(trueTP(2)));
fprintf(1,'est:(%d,%d) ',round(estTP(1)),round(estTP(2)));
deviation = sqrt(sum((estTP-trueTP).^2));
fprintf(1,'distance:  %.1f meters\n',deviation);

    function plotGraphics
        figure(1);
        plot(stackX{1},stackY{1},stackG{1});
        hold on;
        for i=2:stackSize
            plot(stackX{i},stackY{i},stackG{i});
        end
```

```
        axis([594000 596500 4513600 4515600]);
        hold off;
        waitforbuttonpress;
    end

    function [deviation,path] = runway
        depth = 0;
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(1,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(1,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d1,p1]   =   node1(node(1,1)-X(seg(1,2)),node(1,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(2,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(2,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d2,p2]   =   node2(node(2,1)-X(seg(1,2)),node(2,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(3,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(3,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d3,p3]   =   node3(node(3,1)-X(seg(1,2)),node(3,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(4,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(4,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d4,p4]   =   node4(node(4,1)-X(seg(1,2)),node(4,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
```

```
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(5,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(5,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d5,p5]   =   node5(node(5,1)-X(seg(1,2)),node(5,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(22,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(22,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d6,p6] = node22(node(22,1)-X(seg(1,2)),node(22,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(6,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(6,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d7,p7]   =   node6(node(6,1)-X(seg(1,2)),node(6,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        deviation = round([d1;d2;d3;d4;d5;d6;d7]);
        path = [p1;p2;p3;p4;p5;p6;p7];
        path = path(sum(deviation==wrong,2)==0,:);
        deviation = deviation(sum(deviation==wrong,2)==0,:);
        [deviation,path] = select(deviation,path,numSeg-1);
        stackSize = stackSize-1;
    end

    function [deviation,path] =
select(deviation,path,numSeg)
        [temp,I] = min(sum(deviation,2));
        deviation = deviation(I,:);
        path = path(I,:);
    end

    function [deviation,path] =
node1(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
```

```
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(14,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(14,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(14,1)-(X(seg(1,2))+xOff);
        y = node(14,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node14(node(14,1)-
X(seg(1,2)),node(14,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 1;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node2(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(13,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(13,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(13,1)-(X(seg(1,2))+xOff);
        y = node(13,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node13(node(13,1)-X(seg(1,2)),node(13,2)-
Y(seg(1,2)), ...
```

```
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 2;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(20,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(20,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(20,1)-(X(seg(1,2))+xOff);
        y = node(20,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node20(node(20,1)-X(seg(1,2)),node(20,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 2;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node3(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(11,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(11,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(11,1)-(X(seg(1,2))+xOff);
        y = node(11,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
```

```
        [d1,p1] = node11(node(11,1)-X(seg(1,2)),node(11,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 3;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(12,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(12,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(12,1)-(X(seg(1,2))+xOff);
        y = node(12,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node12(node(12,1)-X(seg(1,2)),node(12,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 3;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(20,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(20,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(20,1)-(X(seg(1,2))+xOff);
        y = node(20,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d3,p3] = node20(node(20,1)-X(seg(1,2)),node(20,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d3,1)
            d3(i,depth) = d;
            p3(i,depth) = 3;
        end
        deviation = [d1;d2;d3];
        path = [p1;p2;p3];
        stackSize = stackSize-1;
    end
```

```
    function [deviation,path] =
node4(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(10,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(10,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(10,1)-(X(seg(1,2))+xOff);
        y = node(10,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node10(node(10,1)-X(seg(1,2)),node(10,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 4;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(12,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(12,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(12,1)-(X(seg(1,2))+xOff);
        y = node(12,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node12(node(12,1)-X(seg(1,2)),node(12,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 4;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end
```

```
    function [deviation,path] =
node5(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(9,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(9,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(9,1)-(X(seg(1,2))+xOff);
        y = node(9,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node9(node(9,1)-
X(seg(1,2)),node(9,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 5;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node6(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(7,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(7,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(7,1)-(X(seg(1,2))+xOff);
        y = node(7,2)-(Y(seg(1,2))+yOff);
```

```
        d = sqrt(x^2+y^2);
        [deviation,path] = node7(node(7,1)-
X(seg(1,2)),node(7,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 6;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node7(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(15,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(15,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(15,1)-(X(seg(1,2))+xOff);
        y = node(15,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node15(node(15,1)-X(seg(1,2)),node(15,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 7;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(21,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(21,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(21,1)-(X(seg(1,2))+xOff);
        y = node(21,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
```

```
        [d2,p2] = node21(node(21,1)-X(seg(1,2)),node(21,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 7;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node8(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(15,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(15,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(15,1)-(X(seg(1,2))+xOff);
        y = node(15,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node15(node(15,1)-X(seg(1,2)),node(15,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 8;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(21,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(21,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(21,1)-(X(seg(1,2))+xOff);
        y = node(21,2)-(Y(seg(1,2))+yOff);
```

```
        d = sqrt(x^2+y^2);
        [d2,p2] = node21(node(21,1)-X(seg(1,2)),node(21,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 8;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node9(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(15,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(15,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(15,1)-(X(seg(1,2))+xOff);
        y = node(15,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node15(node(15,1)-X(seg(1,2)),node(15,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 9;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(21,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(21,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(21,1)-(X(seg(1,2))+xOff);
```

```
        y = node(21,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node21(node(21,1)-X(seg(1,2)),node(21,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 9;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node10(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(15,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(15,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(15,1)-(X(seg(1,2))+xOff);
        y = node(15,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node15(node(15,1)-X(seg(1,2)),node(15,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 10;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(21,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(21,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
```

```
        x = node(21,1)-(X(seg(1,2))+xOff);
        y = node(21,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node21(node(21,1)-X(seg(1,2)),node(21,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 10;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node11(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(15,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(15,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(15,1)-(X(seg(1,2))+xOff);
        y = node(15,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node15(node(15,1)-X(seg(1,2)),node(15,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 11;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(21,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(21,2)-
Y(seg(1,2));
```

```
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(21,1)-(X(seg(1,2))+xOff);
        y = node(21,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node21(node(21,1)-X(seg(1,2)),node(21,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 11;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node12(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(15,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(15,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(15,1)-(X(seg(1,2))+xOff);
        y = node(15,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node15(node(15,1)-X(seg(1,2)),node(15,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 12;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(21,1)-
X(seg(1,2));
```

```
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(21,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(21,1)-(X(seg(1,2))+xOff);
        y = node(21,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node21(node(21,1)-X(seg(1,2)),node(21,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 12;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node13(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(15,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(15,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(15,1)-(X(seg(1,2))+xOff);
        y = node(15,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node15(node(15,1)-X(seg(1,2)),node(15,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 13;
        end
```

```
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(21,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(21,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(21,1)-(X(seg(1,2))+xOff);
        y = node(21,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node21(node(21,1)-X(seg(1,2)),node(21,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 13;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node14(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(15,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(15,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(15,1)-(X(seg(1,2))+xOff);
        y = node(15,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path]        =        node15(node(15,1)-
X(seg(1,2)),node(15,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        deviation(1,depth) = d;
        path(1,depth) = 14;
        stackSize = stackSize-1;
```

```
    end

    function [deviation,path] =
node15(xOff,yOff,numSeg,seg,depth)
        if (numSeg~=1)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        deviation = zeros(1,deepest);
        path = zeros(1,deepest);
        path(1,depth) = 15;
    end

    function [deviation,path] =
node16(xOff,yOff,numSeg,seg,depth)
        if ((numSeg==0) | (numSeg>1))
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        deviation = zeros(1,deepest);
        path = zeros(1,deepest);
        path(1,depth) = 16;
    end

    function [deviation,path] =
node17(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(16,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(16,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(16,1)-(X(seg(1,2))+xOff);
        y = node(16,2)-(Y(seg(1,2))+yOff);
```

```
        d = sqrt(x^2+y^2);
        [deviation,path] = node16(node(16,1)-
X(seg(1,2)),node(16,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 17;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node18(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(17,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(17,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(17,1)-(X(seg(1,2))+xOff);
        y = node(17,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node17(node(17,1)-
X(seg(1,2)),node(17,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 18;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node19(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
```

```
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(16,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(16,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(16,1)-(X(seg(1,2))+xOff);
        y = node(16,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node16(node(16,1)-X(seg(1,2)),node(16,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 19;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(18,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(18,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(18,1)-(X(seg(1,2))+xOff);
        y = node(18,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node18(node(18,1)-X(seg(1,2)),node(18,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 19;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node20(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
```

```
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(16,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(16,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(16,1)-(X(seg(1,2))+xOff);
        y = node(16,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node16(node(16,1)-X(seg(1,2)),node(16,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 20;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(18,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(18,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(18,1)-(X(seg(1,2))+xOff);
        y = node(18,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node18(node(18,1)-X(seg(1,2)),node(18,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 20;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node21(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
```

```
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(19,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(19,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(19,1)-(X(seg(1,2))+xOff);
        y = node(19,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node19(node(19,1)-
X(seg(1,2)),node(19,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 21;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node22(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(8,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(8,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(8,1)-(X(seg(1,2))+xOff);
        y = node(8,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node8(node(8,1)-
X(seg(1,2)),node(8,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
```

```
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 22;
        end
        stackSize = stackSize-1;
    end

end
```

A.5. THE FUNCTION THAT PROCESSES LANDING TRACES ON RUNWAY 13.

```
function [deviation,estTP,trueTP] =
predictSE(data,verbosity)

node = [
    595550 4514633; % 1
    595851 4514448; % 2
    596153 4514264; % 3
    596309 4514169; % 4
    596407 4514109; % 5
    595757 4514381; % 6
    595930 4514276; % 7
    595708 4514327; % 8
    595907 4514205; % 9
    596260 4514075; %10
    596033 4514213; %11
    596200 4514026; %12
    596020 4514136; %13
    596343 4514024; %14
    595875 4514033 % parking position
    ];
NWthreshold = [594628.3 4515195.9]; SWthreshold = [594174.7
4513731.8];
SEthreshold = [596450.2 4514082.9]; NEthreshold = [595286.5
4515553.0];
NWdir = acos((SEthreshold(1)-NWthreshold(1)) / ...
    sqrt(sum((NWthreshold-SEthreshold).^2))) * 180/pi + 270;
SEdir = NWdir - 180;
NEdir = 90 - acos((NEthreshold(1)-SWthreshold(1)) / ...
    sqrt(sum((NEthreshold-SWthreshold).^2))) * 180/pi;
SWdir = NEdir + 180;

MHcorrection = 14.2;
wrong = 9999; deepest = 10;
```

```
trueTP = data(1,5:6);
speed = round(data(:,1));
MH = round((data(:,2)-MHcorrection) * 100) / 100;
MH(MH<0) = MH(MH<0)+360;
[X,Y] = cubicUTM(speed,MH);
nw_se = (abs(MH-NWdir)<5.5) | (abs(MH-SEdir)<5.5);
while (1)
    [numSeg,seg] = segment(nw_se);
    again = 0;
    for i=1:numSeg
        if ((seg(i,2)-seg(i,1)+1)<=3)
            nw_se(seg(i,1):seg(i,2)) = abs(nw_se(seg(i,1))-
1);
            again = 1;
        end
    end
    if (again==0) break; end
end

stackX{1} = linspace(NWthreshold(1),SEthreshold(1),100);
stackY{1} = linspace(NWthreshold(2),SEthreshold(2),100);
stackG{1} = '-b';
stackX{2} = linspace(SWthreshold(1),NEthreshold(1),100);
stackY{2} = linspace(SWthreshold(2),NEthreshold(2),100);
stackG{2} = '-b';
stackX{3} = node(:,1); stackY{3} = node(:,2); stackG{3} =
'*b'; stackSize = 3;
if (verbosity==2) plotGraphics; end
stackX{4} = data(:,5); stackY{4} = data(:,6); stackG{4} =
'.k'; stackSize = 4;
if (verbosity>0) plotGraphics; end
for i=1:numSeg
    stackX{i+3} = X(seg(i,1):seg(i,2))+trueTP(1);
    stackY{i+3} = Y(seg(i,1):seg(i,2))+trueTP(2);
    if (mod(i,2)==1) stackG{i+3} = '.r'; else stackG{i+3} =
'.k'; end
end
stackSize = numSeg+3;
if (verbosity>0) plotGraphics; end
stackSize = 3;

% search
[deviation,path] = runway;
% search
```

```
for i=1:(numSeg-1)
    stackX{i+3}   =   X(seg(i,1):seg(i,2))+node(path(1,i),1)-
X(seg(i,2));
    stackY{i+3}   =   Y(seg(i,1):seg(i,2))+node(path(1,i),2)-
Y(seg(i,2));
    stackG{i+3} = '.k';
end
%i = numSeg;
%stackX{i+3} = X(seg(i,1):seg(i,2))+node(end,1)-X(seg(i,2));
%stackY{i+3} = Y(seg(i,1):seg(i,2))+node(end,2)-Y(seg(i,2));
%stackG{i+3} = '.k';
%stackSize = numSeg+3;
stackSize = numSeg+3-1;
for i=1:(numSeg-1)
    stackX{stackSize+i} = node(path(1,i),1);
    stackY{stackSize+i} = node(path(1,i),2);
    stackG{stackSize+i} = '*r';
end
%i = numSeg;
%stackX{stackSize+i} = node(end,1);
%stackY{stackSize+i} = node(end,2);
%stackG{stackSize+i} = '*r';
%stackSize = stackSize+numSeg;
stackSize = stackSize+numSeg-1;
if (verbosity==2) plotGraphics; end

estTP   =   fminsearch(@totalDev,node(path(1),:)-[X(seg(1,2))
Y(seg(1,2))], ...
    [],node(path(1:numSeg-1),:),[X(seg(1:numSeg-1,2))
Y(seg(1:numSeg-1,2))]);

stackX{4} = X+estTP(1);
stackY{4} = Y+estTP(2);
stackG{4} = '.k';
stackSize = 4;
for i=1:(numSeg-1)
    stackX{stackSize+i} = node(path(1,i),1);
    stackY{stackSize+i} = node(path(1,i),2);
    stackG{stackSize+i} = '*r';
end
%i = numSeg;
%stackX{stackSize+i} = node(end,1);
%stackY{stackSize+i} = node(end,2);
%stackG{stackSize+i} = '*r';
%stackSize = stackSize+numSeg;
```

```
stackSize = stackSize+numSeg-1;
if (verbosity==2) plotGraphics; end

coef = polyfit([NWthreshold(1)
SEthreshold(1)]',[NWthreshold(2) SEthreshold(2)]',1);
correction = (estTP(2)-coef(1)*estTP(1)-
coef(2))/(2*coef(1));
estTP(1) = estTP(1)+correction;
estTP(2) = estTP(2)-coef(1)*correction;

stackX{4} = X+estTP(1);
stackY{4} = Y+estTP(2);
stackG{4} = '.k';
stackSize = 4;
for i=1:(numSeg-1)
    stackX{stackSize+i} = node(path(1,i),1);
    stackY{stackSize+i} = node(path(1,i),2);
    stackG{stackSize+i} = '*r';
end
%i = numSeg;
%stackX{stackSize+i} = node(end,1);
%stackY{stackSize+i} = node(end,2);
%stackG{stackSize+i} = '*r';
%stackSize = stackSize+numSeg;
stackSize = stackSize+numSeg-1;
if (verbosity==2) plotGraphics; end

stackX{5} = trueTP(1);
stackY{5} = trueTP(2);
stackG{5} = '*r';
stackSize = 5;
if (verbosity>0) plotGraphics; end

fprintf(1,'GPS:(%d,%d) ',round(trueTP(1)),round(trueTP(2)));
fprintf(1,'est:(%d,%d) ',round(estTP(1)),round(estTP(2)));
deviation = sqrt(sum((estTP-trueTP).^2));
fprintf(1,'distance:  %.1f meters\n',deviation);

    function plotGraphics
        figure(1);
        plot(stackX{1},stackY{1},stackG{1});
        hold on;
        for i=2:stackSize
            plot(stackX{i},stackY{i},stackG{i});
        end
```

```
        axis([594000 596500 4513600 4515600]);
        hold off;
        waitforbuttonpress;
    end

    function [deviation,path] = runway
        depth = 0;
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(1,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(1,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d1,p1]   =   node1(node(1,1)-X(seg(1,2)),node(1,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(2,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(2,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d2,p2]   =   node2(node(2,1)-X(seg(1,2)),node(2,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(3,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(3,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d3,p3]   =   node3(node(3,1)-X(seg(1,2)),node(3,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(4,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(4,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d4,p4]   =   node4(node(4,1)-X(seg(1,2)),node(4,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
```

```
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(5,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(5,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d5,p5]   =   node5(node(5,1)-X(seg(1,2)),node(5,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        deviation = round([d1;d2;d3;d4;d5]);
        path = [p1;p2;p3;p4;p5];
        path = path(sum(deviation==wrong,2)==0,:);
        deviation = deviation(sum(deviation==wrong,2)==0,:);
        [deviation,path] = select(deviation,path,numSeg-1);
        stackSize = stackSize-1;
    end

    function [deviation,path] =
select(deviation,path,numSeg)
        [temp,I] = min(sum(deviation,2));
        deviation = deviation(I,:);
        path = path(I,:);
    end

    function [deviation,path] =
node1(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(6,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(6,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(6,1)-(X(seg(1,2))+xOff);
        y = node(6,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1]   =   node6(node(6,1)-X(seg(1,2)),node(6,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
```

```
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 1;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(8,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(8,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(8,1)-(X(seg(1,2))+xOff);
        y = node(8,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2]   =   node8(node(8,1)-X(seg(1,2)),node(8,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 1;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node2(xOff,yOff,numSeg,seg,depth)
        if ((numSeg==0) | (numSeg>1))
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} =
X(seg(1,1):seg(1,2))+node(end,1)-X(seg(1,2));
        stackY{stackSize} =
Y(seg(1,1):seg(1,2))+node(end,2)-Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(end,1)-(X(seg(1,2))+xOff);
        y = node(end,2)-(Y(seg(1,2))+yOff);
        deviation = zeros(1,deepest);
        path = zeros(1,deepest);
        deviation(1,depth) = sqrt(x^2+y^2);
```

```
        path(1,depth) = 2;
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node3(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(12,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(12,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(12,1)-(X(seg(1,2))+xOff);
        y = node(12,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node12(node(12,1)-
X(seg(1,2)),node(12,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 3;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node4(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(10,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(10,2)-
Y(seg(1,2));
```

```
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(10,1)-(X(seg(1,2))+xOff);
        y = node(10,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node10(node(10,1)-
X(seg(1,2)),node(10,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 4;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node5(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(14,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(14,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(14,1)-(X(seg(1,2))+xOff);
        y = node(14,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node14(node(14,1)-
X(seg(1,2)),node(14,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        deviation(1,depth) = d;
        path(1,depth) = 5;
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node6(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
```

```
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(7,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(7,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(7,1)-(X(seg(1,2))+xOff);
        y = node(7,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node7(node(7,1)-
X(seg(1,2)),node(7,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        deviation(1,depth) = d;
        path(1,depth) = 6;
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node7(xOff,yOff,numSeg,seg,depth)
        if (numSeg~=1)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        deviation = zeros(1,deepest);
        path = zeros(1,deepest);
        path(1,depth) = 7;
    end

    function [deviation,path] =
node8(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
```

```
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(9,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(9,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(9,1)-(X(seg(1,2))+xOff);
        y = node(9,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node9(node(9,1)-
X(seg(1,2)),node(9,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        deviation(1,depth) = d;
        path(1,depth) = 8;
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node9(xOff,yOff,numSeg,seg,depth)
        if (numSeg~=1)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        deviation = zeros(1,deepest);
        path = zeros(1,deepest);
        path(1,depth) = 9;
    end

    function [deviation,path] =
node10(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(11,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(11,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
```

```
        x = node(11,1)-(X(seg(1,2))+xOff);
        y = node(11,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node11(node(11,1)-
X(seg(1,2)),node(11,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        deviation(1,depth) = d;
        path(1,depth) = 10;
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node11(xOff,yOff,numSeg,seg,depth)
        if (numSeg~=1)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        deviation = zeros(1,deepest);
        path = zeros(1,deepest);
        path(1,depth) = 11;
    end

    function [deviation,path] =
node12(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(13,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(13,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(13,1)-(X(seg(1,2))+xOff);
        y = node(13,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path]        =        node13(node(13,1)-
X(seg(1,2)),node(13,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
```

```
        deviation(1,depth) = d;
        path(1,depth) = 12;
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node13(xOff,yOff,numSeg,seg,depth)
        if (numSeg~=1)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        deviation = zeros(1,deepest);
        path = zeros(1,deepest);
        path(1,depth) = 13;
    end

    function [deviation,path] =
node14(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(11,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(11,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(11,1)-(X(seg(1,2))+xOff);
        y = node(11,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node11(node(11,1)-
X(seg(1,2)),node(11,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        deviation(1,depth) = d;
        path(1,depth) = 14;
        stackSize = stackSize-1;
    end

end
```

A.6. THE FUNCTION THAT PROCESSES LANDING TRACES ON RUNWAY 22.

```
function [deviation,estTP,trueTP] =
predictSW(data,verbosity)

node = [
    594716 4514620; ... % 1
    594657 4514524; ... % 2
    594541 4514332; ... % 3
    594321 4513974; ... % 4
    594199 4513773; ... % 5
    594412 4513917; ... % 6
    594594 4514213; ... % 7
    594746 4514462; ... % 8
    594803 4514557; ... % 9
    594848 4514630; ... %10
    594856 4514526; ... %11
    594916 4514629; ... %12
    595151 4514680; ... %13
    595533 4514447; ... %14
    595582 4514403; ... %15
    595946 4514181; ... %16
    595170 4514744; ... %17
    595944 4514272; ... %18
    594884 4514690; ... %19
    595875 4514033 % parking position
    ];
NWthreshold = [594628.3 4515195.9]; SWthreshold = [594174.7
4513731.8];
SEthreshold = [596450.2 4514082.9]; NEthreshold = [595286.5
4515553.0];
NWdir = acos((SEthreshold(1)-NWthreshold(1)) / ...
    sqrt(sum((NWthreshold-SEthreshold).^2))) * 180/pi + 270;
SEdir = NWdir - 180;
NEdir = 90 - acos((NEthreshold(1)-SWthreshold(1)) / ...
    sqrt(sum((NEthreshold-SWthreshold).^2))) * 180/pi;
SWdir = NEdir + 180;
MHcorrection = 14.2;
wrong = 9999; deepest = 10;

trueTP = data(1,5:6);
speed = round(data(:,1));
MH = round((data(:,2)-MHcorrection) * 100) / 100;
MH(MH<0) = MH(MH<0)+360;
[X,Y] = cubicUTM(speed,MH);
```

```
sw_ne = (abs(MH-SWdir)<5.5) | (abs(MH-NEdir)<5.5);
while (1)
    [numSeg,seg] = segment(sw_ne);
    again = 0;
    for i=1:numSeg
        if ((seg(i,2)-seg(i,1)+1)<=4)
            sw_ne(seg(i,1):seg(i,2)) = abs(sw_ne(seg(i,1))-
1);
            again = 1;
        end
    end
    if (again==0) break; end
end
sw_ne(seg(numSeg-1,1):seg(numSeg-1,2)) = 0;
if (sw_ne(1) ~= 1) fprintf(1,'%d wrong\n'); end
nw_se = (abs(MH-NWdir)<5.5) | (abs(MH-SEdir)<5.5);
nw_se(1:seg(3,1)) = 0;
all = (sw_ne | nw_se);
while (1)
    [numSeg,seg] = segment(all);
    again = 0;
    for i=1:numSeg
        if ((seg(i,2)-seg(i,1)+1)<=4)
            all(seg(i,1):seg(i,2)) = abs(all(seg(i,1))-1);
            again = 1;
        end
    end
    if (again==0) break; end
end

stackX{1} = linspace(NWthreshold(1),SEthreshold(1),100);
stackY{1} = linspace(NWthreshold(2),SEthreshold(2),100);
stackG{1} = '-b';
stackX{2} = linspace(SWthreshold(1),NEthreshold(1),100);
stackY{2} = linspace(SWthreshold(2),NEthreshold(2),100);
stackG{2} = '-b';
stackX{3} = node(:,1); stackY{3} = node(:,2); stackG{3} =
'*b'; stackSize = 3;
if (verbosity==2) plotGraphics; end
stackX{4} = data(:,5); stackY{4} = data(:,6); stackG{4} =
'.k'; stackSize = 4;
if (verbosity>0) plotGraphics; end
for i=1:numSeg
    stackX{i+3} = X(seg(i,1):seg(i,2))+trueTP(1);
    stackY{i+3} = Y(seg(i,1):seg(i,2))+trueTP(2);
```

```
    if (mod(i,2)==1) stackG{i+3} = '.r'; else stackG{i+3} =
'.k'; end
end
stackSize = numSeg+3;
if (verbosity>0) plotGraphics; end
stackSize = 3;

% search
[deviation,path] = runway;
% search

for i=1:(numSeg-1)
    stackX{i+3}   =   X(seg(i,1):seg(i,2))+node(path(1,i),1)-
X(seg(i,2));
    stackY{i+3}   =   Y(seg(i,1):seg(i,2))+node(path(1,i),2)-
Y(seg(i,2));
    stackG{i+3} = '.k';
end
%i = numSeg;
%stackX{i+3} = X(seg(i,1):seg(i,2))+node(end,1)-X(seg(i,2));
%stackY{i+3} = Y(seg(i,1):seg(i,2))+node(end,2)-Y(seg(i,2));
%stackG{i+3} = '.k';
%stackSize = numSeg+3;
stackSize = numSeg+3-1;
for i=1:(numSeg-1)
    stackX{stackSize+i} = node(path(1,i),1);
    stackY{stackSize+i} = node(path(1,i),2);
    stackG{stackSize+i} = '*r';
end
%i = numSeg;
%stackX{stackSize+i} = node(end,1);
%stackY{stackSize+i} = node(end,2);
%stackG{stackSize+i} = '*r';
%stackSize = stackSize+numSeg;
stackSize = stackSize+numSeg-1;
if (verbosity==2) plotGraphics; end

estTP   =   fminsearch(@totalDev,node(path(1),:)-[X(seg(1,2))
Y(seg(1,2))], ...
    [],node(path(1:numSeg-1),:),[X(seg(1:numSeg-1,2))
Y(seg(1:numSeg-1,2))]);

stackX{4} = X+estTP(1);
stackY{4} = Y+estTP(2);
stackG{4} = '.k';
```

```
stackSize = 4;
for i=1:(numSeg-1)
    stackX{stackSize+i} = node(path(1,i),1);
    stackY{stackSize+i} = node(path(1,i),2);
    stackG{stackSize+i} = '*r';
end
%i = numSeg;
%stackX{stackSize+i} = node(end,1);
%stackY{stackSize+i} = node(end,2);
%stackG{stackSize+i} = '*r';
%stackSize = stackSize+numSeg;
stackSize = stackSize+numSeg-1;
if (verbosity==2) plotGraphics; end

coef = polyfit([NEthreshold(1)
SWthreshold(1)]',[NEthreshold(2) SWthreshold(2)]',1);
correction = (estTP(2)-coef(1)*estTP(1)-
coef(2))/(2*coef(1));
estTP(1) = estTP(1)+correction;
estTP(2) = estTP(2)-coef(1)*correction;

stackX{4} = X+estTP(1);
stackY{4} = Y+estTP(2);
stackG{4} = '.k';
stackSize = 4;
for i=1:(numSeg-1)
    stackX{stackSize+i} = node(path(1,i),1);
    stackY{stackSize+i} = node(path(1,i),2);
    stackG{stackSize+i} = '*r';
end
%i = numSeg;
%stackX{stackSize+i} = node(end,1);
%stackY{stackSize+i} = node(end,2);
%stackG{stackSize+i} = '*r';
%stackSize = stackSize+numSeg;
stackSize = stackSize+numSeg-1;
if (verbosity==2) plotGraphics; end

stackX{5} = trueTP(1);
stackY{5} = trueTP(2);
stackG{5} = '*r';
stackSize = 5;
if (verbosity>0) plotGraphics; end

fprintf(1,'GPS:(%d,%d) ',round(trueTP(1)),round(trueTP(2)));
```

```
fprintf(1,'est:(%d,%d) ',round(estTP(1)),round(estTP(2)));
deviation = sqrt(sum((estTP-trueTP).^2));
fprintf(1,'distance:  %.1f meters\n',deviation);

    function plotGraphics
        figure(1);
        plot(stackX{1},stackY{1},stackG{1});
        hold on;
        for i=2:stackSize
            plot(stackX{i},stackY{i},stackG{i});
        end
        axis([594000 596500 4513600 4515600]);
        hold off;
        waitforbuttonpress;
    end

    function [deviation,path] = runway
        depth = 0;
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(1,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(1,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d1,p1]   =   node1(node(1,1)-X(seg(1,2)),node(1,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(2,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(2,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d2,p2]   =   node2(node(2,1)-X(seg(1,2)),node(2,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(3,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(3,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d3,p3]   =   node3(node(3,1)-X(seg(1,2)),node(3,2)-
Y(seg(1,2)), ...
```

```
            numSeg-1,seg(2:end,:),depth+1);
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(4,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(4,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d4,p4]   =   node4(node(4,1)-X(seg(1,2)),node(4,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(5,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(5,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        [d5,p5]   =   node5(node(5,1)-X(seg(1,2)),node(5,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        deviation = round([d1;d2;d3;d4;d5]);
        path = [p1;p2;p3;p4;p5];
        path = path(sum(deviation==wrong,2)==0,:);
        deviation = deviation(sum(deviation==wrong,2)==0,:);
        [deviation,path] = select(deviation,path,numSeg-1);
        stackSize = stackSize-1;
    end

    function [deviation,path] =
select(deviation,path,numSeg)
        [temp,I] = min(sum(deviation,2));
        deviation = deviation(I,:);
        path = path(I,:);
    end

    function [deviation,path] =
node1(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(9,1)-
X(seg(1,2));
```

```
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(9,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(9,1)-(X(seg(1,2))+xOff);
        y = node(9,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1]   =   node9(node(9,1)-X(seg(1,2)),node(9,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 1;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(11,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(11,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(11,1)-(X(seg(1,2))+xOff);
        y = node(11,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node11(node(11,1)-X(seg(1,2)),node(11,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 1;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node2(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
```

```
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(8,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(8,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(8,1)-(X(seg(1,2))+xOff);
        y = node(8,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1]   =   node8(node(8,1)-X(seg(1,2)),node(8,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 2;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(11,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(11,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(11,1)-(X(seg(1,2))+xOff);
        y = node(11,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node11(node(11,1)-X(seg(1,2)),node(11,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 2;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node3(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
```

```
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(7,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(7,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(7,1)-(X(seg(1,2))+xOff);
        y = node(7,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node7(node(7,1)-
X(seg(1,2)),node(7,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 3;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node4(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(6,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(6,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(6,1)-(X(seg(1,2))+xOff);
        y = node(6,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node6(node(6,1)-
X(seg(1,2)),node(6,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 4;
        end
```

```
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node5(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(6,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(6,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(6,1)-(X(seg(1,2))+xOff);
        y = node(6,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node6(node(6,1)-
X(seg(1,2)),node(6,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 5;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node6(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(8,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(8,2)-
Y(seg(1,2));
```

```
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(8,1)-(X(seg(1,2))+xOff);
        y = node(8,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1]   =   node8(node(8,1)-X(seg(1,2)),node(8,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 6;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(10,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(10,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(10,1)-(X(seg(1,2))+xOff);
        y = node(10,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node10(node(10,1)-X(seg(1,2)),node(10,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 6;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(19,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(19,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(19,1)-(X(seg(1,2))+xOff);
        y = node(19,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d3,p3] = node19(node(19,1)-X(seg(1,2)),node(19,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d3,1)
            d3(i,depth) = d;
            p3(i,depth) = 6;
        end
        deviation = [d1;d2;d3];
```

```
        path = [p1;p2;p3];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node7(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(8,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(8,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(8,1)-(X(seg(1,2))+xOff);
        y = node(8,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1]   =   node8(node(8,1)-X(seg(1,2)),node(8,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 7;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(10,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(10,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(10,1)-(X(seg(1,2))+xOff);
        y = node(10,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node10(node(10,1)-X(seg(1,2)),node(10,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 7;
        end
```

```
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(19,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(19,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(19,1)-(X(seg(1,2))+xOff);
        y = node(19,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d3,p3] = node19(node(19,1)-X(seg(1,2)),node(19,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d3,1)
            d3(i,depth) = d;
            p3(i,depth) = 7;
        end
        deviation = [d1;d2;d3];
        path = [p1;p2;p3];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node8(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(11,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(11,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(11,1)-(X(seg(1,2))+xOff);
        y = node(11,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node11(node(11,1)-X(seg(1,2)),node(11,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 8;
```

```
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(10,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(10,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(10,1)-(X(seg(1,2))+xOff);
        y = node(10,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node10(node(10,1)-X(seg(1,2)),node(10,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 8;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(19,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(19,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(19,1)-(X(seg(1,2))+xOff);
        y = node(19,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d3,p3] = node19(node(19,1)-X(seg(1,2)),node(19,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d3,1)
            d3(i,depth) = d;
            p3(i,depth) = 8;
        end
        deviation = [d1;d2;d3];
        path = [p1;p2;p3];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node9(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
```

```
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(10,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(10,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(10,1)-(X(seg(1,2))+xOff);
        y = node(10,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node10(node(10,1)-X(seg(1,2)),node(10,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 9;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(19,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(19,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(19,1)-(X(seg(1,2))+xOff);
        y = node(19,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node19(node(19,1)-X(seg(1,2)),node(19,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 9;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node10(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
```

```
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(13,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(13,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(13,1)-(X(seg(1,2))+xOff);
        y = node(13,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node13(node(13,1)-
X(seg(1,2)),node(13,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 10;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node11(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(12,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(12,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(12,1)-(X(seg(1,2))+xOff);
        y = node(12,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node12(node(12,1)-
X(seg(1,2)),node(12,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
```

```
            path(i,depth) = 11;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node12(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(13,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(13,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(13,1)-(X(seg(1,2))+xOff);
        y = node(13,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node13(node(13,1)-
X(seg(1,2)),node(13,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 12;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node13(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(14,1)-
X(seg(1,2));
```

```
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(14,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(14,1)-(X(seg(1,2))+xOff);
        y = node(14,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d1,p1] = node14(node(14,1)-X(seg(1,2)),node(14,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d1,1)
            d1(i,depth) = d;
            p1(i,depth) = 13;
        end
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(16,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(16,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(16,1)-(X(seg(1,2))+xOff);
        y = node(16,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [d2,p2] = node16(node(16,1)-X(seg(1,2)),node(16,2)-
Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(d2,1)
            d2(i,depth) = d;
            p2(i,depth) = 13;
        end
        deviation = [d1;d2];
        path = [p1;p2];
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node14(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
```

```
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(15,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(15,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(15,1)-(X(seg(1,2))+xOff);
        y = node(15,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node15(node(15,1)-
X(seg(1,2)),node(15,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 14;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node15(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(16,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(16,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(16,1)-(X(seg(1,2))+xOff);
        y = node(16,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node16(node(16,1)-
X(seg(1,2)),node(16,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 15;
        end
        stackSize = stackSize-1;
```

```
    end

    function [deviation,path] =
node16(xOff,yOff,numSeg,seg,depth)
        if (numSeg~=1)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        deviation = zeros(1,deepest);
        path = zeros(1,deepest);
        path(1,depth) = 16;
    end

    function [deviation,path] =
node17(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(18,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(18,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(18,1)-(X(seg(1,2))+xOff);
        y = node(18,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node18(node(18,1)-
X(seg(1,2)),node(18,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 17;
        end
        stackSize = stackSize-1;
    end

    function [deviation,path] =
node18(xOff,yOff,numSeg,seg,depth)
```

```
        if (numSeg~=1)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        deviation = zeros(1,deepest);
        path = zeros(1,deepest);
        path(1,depth) = 18;
    end

    function [deviation,path] =
node19(xOff,yOff,numSeg,seg,depth)
        if (numSeg==0)
            deviation = zeros(1,deepest);
            path = zeros(1,deepest);
            deviation(1,depth) = wrong;
            return;
        end
        stackSize = stackSize+1;
        stackX{stackSize} = X(seg(1,1):seg(1,2))+node(17,1)-
X(seg(1,2));
        stackY{stackSize} = Y(seg(1,1):seg(1,2))+node(17,2)-
Y(seg(1,2));
        stackG{stackSize}   =   '.k';   if   (verbosity==2)
plotGraphics; end
        x = node(17,1)-(X(seg(1,2))+xOff);
        y = node(17,2)-(Y(seg(1,2))+yOff);
        d = sqrt(x^2+y^2);
        [deviation,path] = node17(node(17,1)-
X(seg(1,2)),node(17,2)-Y(seg(1,2)), ...
            numSeg-1,seg(2:end,:),depth+1);
        for i=1:size(deviation,1)
            deviation(i,depth) = d;
            path(i,depth) = 19;
        end
        stackSize = stackSize-1;
    end

end
```

A.7. THE FUNCTION THAT CALCULATES THE DISTANCE BETWEEN THE SEGMENTS OF THE LANDING TRACES AND THE RED MARKS ON THE RUNWAY/TAXIWAY.

```
function d = totalDev(v,node,trace)
d = sum(sqrt((trace(:,1)+v(1)-node(:,1)).^2 +
(trace(:,2)+v(2)-node(:,2)).^2));
```