



# Analyzing Distances Based on Pixels using MATLAB: An Image Analysis Study



Sourav Das, Ryan Nelson, Thomas Cumming and Cody William Casper\*

Mechanical Engineering Department, University of Wisconsin Milwaukee College of Engineering & Applied Science, USA

Submission: August 08, 2017; Published: August 23, 2018

\*Corresponding author: Cody William Casper, Mechanical Engineering Department, University of Wisconsin Milwaukee College of Engineering & Applied Science 53201-0784, Milwaukee, USA. Email: ccasper@uwm.edu

## Abstract

The present research paper is addressed to the distance between major US airports using image analysis to give the user an accurate distance and estimated flight time for all major US airports. The system inputs needed are an average distance between the different US airports as well as the average flight speed. This research is to utilize pixels from an image to create a coordinate system, which allows us to calculate distances. Conversion factors will also be included in our system inputs. The user inputs for our program will be your starting location and the destination you are looking to travel to. The governing equations for our code are the system input coordinates on our image mapping and converting the distance between the coordinates to an actual distance in miles. These coordinates and the use of the equation of a line to get an accurate distance between airports in miles. Multiplying the total distance by the average plane speed, while accounting for taxi times, we got an accurate representation of the time it will take to get between airports. The user-defined functions are a multi-choice dialog box that the user is required to click on the location that they are starting at which leads to another multi choice dialog box requiring the user to input the destination they are looking to go to. The entire bases of our research were based on menu and switch cases.

**Keywords:** Pixels; US airports; Flight information; Matlab; Distance; Image analysis

## Introduction

The distance between major US airports using image analysis to give the user an accurate distance and estimated flight time for all major US airports. The system inputs needed are an average distance between the different US airports as well as the average flight speed [1]. Conversion factors will also be included in system inputs. The user inputs for our program will be your starting location and the destination you are looking to travel to. The governing equations code are the system input coordinates on image mapping and converting the distance between the coordinates to an actual distance in miles. These coordinates and use of the equation of a line to get an accurate distance between airports in miles. Multiplying the total distance by the average plane speed, while accounting for taxi times, one can get an accurate representation of the time it will take to get between airports.

The user-defined functions are a multi-choice dialog box that the user is required to click on the location that they are starting at which leads to another multi choice dialog box requiring the user to input the destination they are looking to go to. The entire bases of this research article will be based on menu and switch cases. For example, if the user selects LAX, system use a switch case for the different combinations of airports then display the information desired by the user. To validate our program, the program can select the airport in which we want to go to. It

should tell us the distance and approximate time it will take to reach the destination. We can verify our results by using online flight information including the actual distance and time and compare them to our code.

## Methods and Overview Software structure GUI structures

```
%% pulling up the image

clear Dis;
clear time;
clc;

%leave the image to your computer and copy the path from the file folder
%into the path below
%path='this is where you would enter your path'

path='I:\Mech 101\Project';
airports=imread('airports.jpg');
imshow(airports);
```

Figure 1: Pulling up the image.

The structure which was used for the code was sectioned based, so we could define which part of the code does what in a clear way. The first section of the code pulls up the image that was used to define where the airports, we analyzed, are on a map of the United States [2]. The commands “clear dis, clear time, and clc” were used to clear any previous distance and time calculations in the workspace and command windows to avoid confusion. After the command codes were run we told MATLAB

where our image is located within the computer by creating a path using the image file address. It was necessary to use the command imread to allow MATLAB to read the image from a graphics file (Figure 1).

```
%% Define Pixel Length
% Define pixel length by using the distance from San Francisco to Los
% Angeles which is 347 miles. The equation used to find this is using the
% equation of a line to find the pixels between airports, then dividing the
% distance (347) by the pixels to define our pixel length, which is
% 5.35 miles per pixel.

SANLAX=347;
px1=90;
py1=141;
px2=57;
py2=200;
PR=sqrt(((px2-px1).^2)+(py2-py1).^2);
mpp=SANLAX/PR;
%disp(mpp)
```

Figure 2: Defining pixel Length.

By finding the actual distance between the San Francisco (SAN) and Los Angeles (LAX) Airports (approximately 347 miles) we were able to define the length, in miles, that each pixel in the image represented. By finding the centroids of both SAN and LAX represented by circles on the image and then manually inputting those coordinates into our script. Applying those coordinates in the equation of a line we were able to find the pixel length between SAN and LAX. The equation of a line is in Appendix

B. We discovered how many miles each pixel represented by utilizing how many pixels were present between SAN and LAX, dividing out the actual distance in miles by the pixel distance to give us our miles per pixel(map) (Figure 2).

```
%% Departure airport
% Creates a menu to select departure airport from
% Gives x1 and y1 coordinates for equation of a line

dairport=menu('Select Departure airport: ','Seattle(SEA)')
switch dairport
    case 1;
        disp('Seattle(SEA)');
        x1=77;
        y1=19;
    case 2;
        disp('San Francisco(SFO)');
        x1=90;
        y1=141;
```

Figure 3: Switch Case apply.

To adjust our code to a user-friendly platform, the centroids of the top twenty US airports where represented as x and y coordinates which we hard-coded into a switch case menu. Then creating a second similar menu, so the user could select both a departure and arrival airport from separate menus while giving us two different x and y coordinates for each airport.

```
%% Distance between airports
% Calculates distance between airports from selected coordinates from the
% equation of a line
% Converts to miles by converting pixels to distance in mile or miles per
% pixel (mpp)
% Calculating Time
% the adverage flight speed is 575 MPH
% the adverage taxi time is 16 mins
% Gives average flight time by dividing the calculated distance between
% airports and the adverage flight speed
% the if statement will not allow you to select the same departure and
% arrival airport

format short
if dairport==airport;
    disp('Departure airport and Arival airport can not be the same')
else
    l=sqrt(((x2-x1).^2)+(y2-y1).^2);
    Dis=l*mpp;
    fprintf('The distance between the airports is %g miles',Dis)

    mph=575;
    time=32/60;
    ftime=Dis/mph;
    time=ftime+time;
    fprintf(' and the flight time between the airports is %1.3g hours',time)
end
```

Figure 4: Distance between Airports.

In Figure 3 we employed the selected airport’s coordinates into an if else statement which allowed us to create a case where the code would not work if the user selected the same departure and arrival airport. If the chosen airports where different, the coordinates are then placed into the equation of a line to find the distance in pixels between those two airports. To convert the distance from pixels to miles we multiplied our pixels by the miles per pixel(mpp) we found earlier in our code. The average flight time was found by dividing that distance in miles by the average flight speed (approximately 575 mph) and adding the average taxi times (approximately 32 minutes) to that found number.

Displaying our results in Figure 4 to the user by using fprintf statements to print the results into the command window. Finally, to display on the map where the user would be traveling to and

from, we used the command “line” to draw a line on our image between the two selected airports. The result of this code is a statement clearly defining what the estimated distance and flight time between two airports that the user selects from the menus as well as a line on the map to display a visual representation of the estimated flight path (Figure 5).

```
%% Draw a line on the map between airports
% Draws a line on the map between defined airports using selected
% coordinates from the menu

line([x1,x2],[y1,y2],'Color','r','LineWidth',1)
```

Figure 5: Defining the estimated flight paths .

Result and Discussion

To create a coordinate system, which allows us to calculate distances. We did this by using an equation of a line and a picture, which gave us the ability to convert the number of miles in one

pixel. This kind of program was created to provide a simple interface for a US traveller to quickly find the distance and time for an airline flight from their departure and arrival airports. We were able to achieve this by using multiple Matlab tools, such as imread, show, switch cases, if else statements, etc (Figures 6-8).

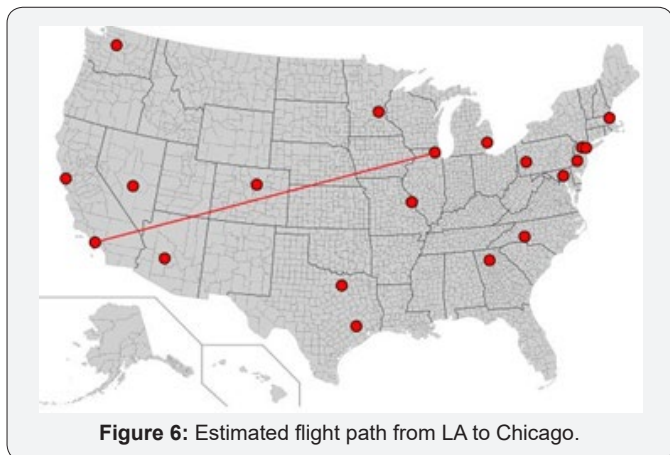


Figure 6: Estimated flight path from LA to Chicago.

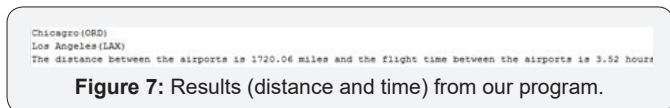


Figure 7: Results (distance and time) from our program.

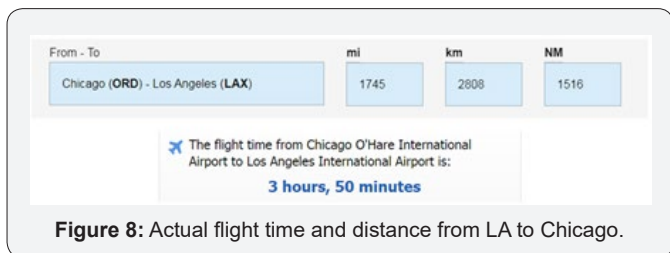


Figure 8: Actual flight time and distance from LA to Chicago.

There are multiple ways that we can make the program better and more efficient. One way is to utilize better equations, for example, we used an average flight speed; instead of this, we could introduce a fluctuating speed through differential equations. A better way to calculate the flight path would be by

using flight path equations with drag. Another way we could make the program more accurate is to use a higher quality image. If the image were of higher quality, then the coordinate accuracy would be much more precise. One thing that could be taken into consideration is more variables such as wind speed, drag, and taxi times. If these variables are introduced, then the times and distances can be calculated much more precisely. If all these improvements get put into the project, then our program would run much more efficiently and accurately.


There are multiple ways that we can make the program better and more efficient. One way is to utilize better equations, for example, we used an average flight speed; instead of this, we could introduce a fluctuating speed through differential equations. A better way to calculate the flight path would be by using flight path equations with drag. Another way we could make the program more accurate is to use a higher quality image. If the image were of higher quality, then the coordinate accuracy would be much more precise. One thing that could be taken into consideration is more variables such as wind speed, drag, and taxi times. If these variables are introduced, then the times and distances can be calculated much more precisely. If all these improvements get put into the project, then our program would run much more efficiently and accurately.

### Acknowledgment

Authors thank Dr. Jenna E Pink for giving ideas of using pixels as a unit for measuring distances. Authors also thank the Department of Mechanical Engineering, the University of Wisconsin-Milwaukee for providing necessary facilities to work in this area as a part of the undergraduate project.

### References

1. Distance from Chicago (ORD) to Los Angeles (LAX)." Air Miles Calculator.
2. Flight Time from ORD to LAX." Travel math.

 This work is licensed under Creative Commons Attribution 4.0 License  
DOI: 10.19080/ETOAJ.2018.02.555587

#### Your next submission with Juniper Publishers will reach you the below assets

- Quality Editorial service
- Swift Peer Review
- Reprints availability
- E-prints Service
- Manuscript Podcast for convenient understanding
- Global attainment for your research
- Manuscript accessibility in different formats  
( Pdf, E-pub, Full Text, Audio)
- Unceasing customer service

Track the below URL for one-step submission  
<https://juniperpublishers.com/online-submission.php>