# Introduction

This tutorial tells you how the following map files are made:

Edge.csv

posName.csv

LongLatPos.csv

For one airport, it takes the 3 files shown above and a ‘naïve\_graph.csv’ file to completely define an airport map. They are used in such way:

Waypoints

ATC Command

airtraceViewer

Edge.csv

posName.csv

LongLatPos.csv

naive\_graph.csv

Figure 0.

To learn more about ‘naive\_graph.csv’ please read ‘WhatIsCompleteGraph.pdf’. Note that the MATLAB code described in that document has now been implemented in Python. To know about the other 3 .csv files continue read this document.

# naïve\_graph.csv

This file defines a graph of the airport layout, with input/output points to the taxiway environment and taxiway intersections named. It also defines and names taxiways between these points. An example is shown in Figure 1 below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C1 | Input/Output | CB via C |  |  |  |
| A1 | Input/Output | CA via A1 |  |  |  |
| A2 | Input/Output | BA via A2 |  |  |  |
| A3 | Input/Output | BA via A |  |  |  |
| CB | Intersection | C1 via C | BA via B | CA via C |  |
| BA | Intersection | A2 via A2 | A3 via A | CB via B | CA via A |
| CA | Intersection | A1 via A1 | CB via C | BA via A |  |

Figure 1

The first entry in each row of the csv file is the name of the node. The next entry is its type – either an Input/Output point, or a taxiway intersection. Finally, any remaining entries in the row represent outedges from this node. They are described in the format “[destination node] via [taxiway name]”.

# Edge.csv & LongLatPos.csv

First, see the picture bellow.

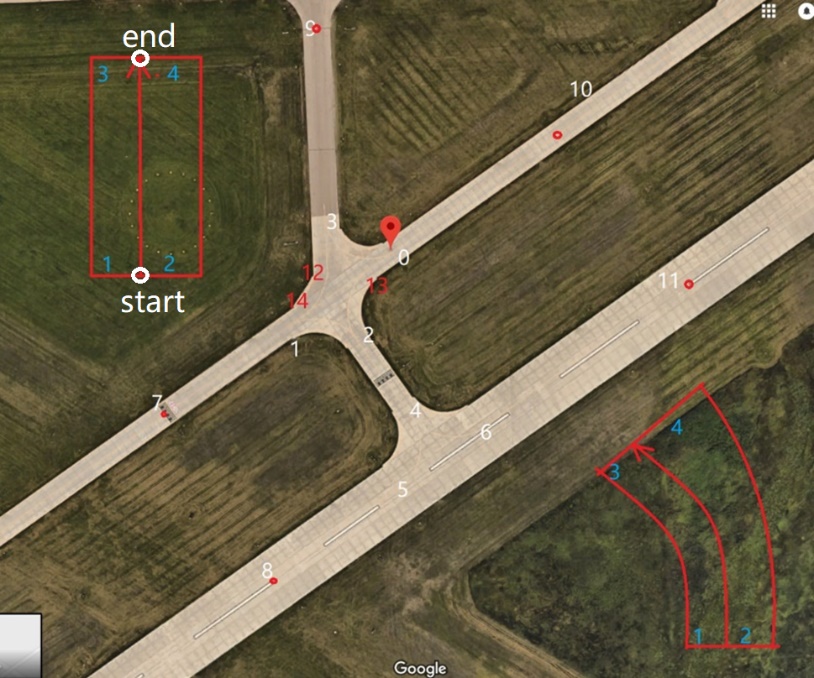


Figure 2.

It is from Ann Arbor Municipal Airport of Google map. The road pieces of a airport can be always segmented into finite small straight pieces and curved pieces with constant curvature. We define a road piece by telling the following information of it:

: a index representing the starting point of the road piece’s center/yellow line.

: a index representing the ending point of the road piece’s center/yellow line.

As is shown in Figure 2, there are many points marked with indexes in the picture (from 0 to 11 in the picture). They are serving as the and points of road pieces. In **LongLatPos.csv** you can see a (n,2) matrix with being the longitude and latitude value of the point in Figure 1.( starts from 0).

: the curvature of the road piece (in ), where means if you go from to you will be turning left.

: The road piece’s width(in ). As can be seen from Figure 2 if you go from point 4 to point 5, the width of the road is getting bigger, and the center/yellow line is moving from the middle of the road to its left. That’s why there are 4 different lengths needed to better determine the shape of the road, like the distance from to the left road side is . The same goes for to whose correspondence are depicted in Figure 2 by the red boxes.

: a integer specifying road type:

0:"Straight Taxi",

1:"Curved Taxi",

2:"Striaght Intersection",

3: "Curved Intersection",

4:"Striaght Runway",

5: "Curved Runway"

Current priority: If only a piece is consisting a intersection it’s type should be 2 or 3, otherwise judge whether it’s overlapping a run way and make it 4 or 5 if so.

The content of **Edge.csv** is simply a matrix with each row of it a vector defining a road piece. If then and are considered ‘connected’. The order of the ’s doesn’t matter.

# posName.csv

In this file, each row is a pair :(, ).

where is the index of a point in Figure 2, and is the ‘node name’ of that point inside **complete\_graph\_autogen.csv** . To see what I mean, go thru ‘WhatIsCompleteGraph.pdf’, which is a copy of Guillaume’s final report.