**Ethan Glassman**

**User Guide**

**October 6 2013**

**Personalized Trip Maps**

**Introduction:**

This project uses a pair of python scripts written to extract your personal gps tracking data recorded from a cellphone running **android**, and generate a personalized, physical map of your trip. This data is collected passively if you have the **Access to my location** setting turned on on your android device, and can be accessed at:

**https://maps.google.com/locationhistory/b/0**

Because this data is collected passively, you can make these maps of any interesting trip you have made at any time in the past, as well as planned future trips.

These maps can make great mementoes, by using data that is passively collected you can enjoy your trip as it happens, and generate a map once you are comfortably back home.

Please feel free to contact me at [efinkg@gmail.com](mailto:efinkg@gmail.com) if you have any questions, difficulties in using these scripts or trouble locating the proper python libraries.

**Assumed competence levels:**

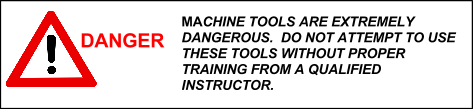
Software:

* Downloading and installing new python libraries

*Note: You will need to be fairly comfortable installing python libraries, since a number of the required packages are relatively old and may have to be built from source.*

* Modifying and running python scripts
* Importing files into a CAD package (I use Solidworks)
* Extruding solid models

*Note: the following operations could be done by a 3rd party manufacturer if you don’t have the necessary experience, safety training, and access to CNC machines to do the engraving yourself.*



* Using the engraving packages on your preferred CAM software (I use HSMWorks)
* Generating gcode to machine parts

Hardware:

* Machining complex shapes in at least 2.5d

**Software packages you will need to have:**

**Python**: The computational heavy lifting converting from a KML file to a PostScript file is done in python.

*Note: If you don’t already have python installed on your system this is likely not a good project for you.*

You will need the following packages installed:

* **pkyKML** for importing data from your KML file. pyKML has the following additional requirements to install before pyKML:
  + **lxml**
  + **libxml2**
  + **libxslt**
* **numpy**
* **matplotlib**
* **basemap**

**pstoedit** is used for converting the PostScript file to a dxf document that will be able to imported to **your preferred 3D CAD software.** Both pstoedit and Solidworks only run on Windows. If you are not planning to make and machine your own 3d version of this map, however, python and the python packages you will be using are a relatively platform agnostic tool.

**Overview of the Map Generation Process**

1. Clone code from **github.**
2. Download your personal tracking data in KML from Google.
3. Preprocess the data, removing ‘gx:’ from kml tags **in your preferred text editor.**
4. Use **MapUnique.py** to generate a latitude longitude list in csv that can be plotted.
5. Write a csv file of any stops you want to display.
6. Use ***USMapFromLat.py*** to generate your map in PostScript format..
7. Use **pstoedit** to convert your PostScript map to a dxf file.
8. Import your map in dxf to **your preferred 3D CAD package.**
9. Clean up your dxf and extrude to a solid.
10. Use **your preferred CAM package** to prepare gcode.
11. Machine the map.

**1. Clone code from github:**

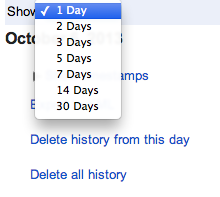
*Note: If this sentence doesn’t mean anything to you, you are probably in over your head for the rest of the process.*

For this step, download the python scripts from <https://github.com/efinkg/Map>.

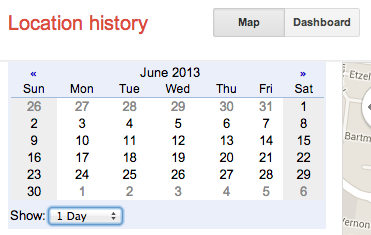
The python scripts, as well as a couple of example files from a trip I took are located on github. You will definitely need **MapUnique.py** and **USMapFromLat.py**, the rest of the files are examples or old versions of code which may be instructive to look at if you are curious about the forms various input and output files will have.

**2. Download your personal tracking data in KML from Google:**

To download the data you will be using go to [**https://maps.google.com/locationhistory/b/0**](https://maps.google.com/locationhistory/b/0), select the length of time you would like to make a map from from the dropdown menu:



Select the start date from the calendar above the dropdown menu:



If you want more than 30 days worth of data download multiple data sets and combine your resulting csv files after step 4. If you want a different length of time you can further manipulate your kml file in a text editor in step 3.

**3. Preprocess the data, removing ‘gx:’ from kml tags in your preferred text editor.**

Before you will be able to use the python scripts you will need to do some slight preprocessing.

In this step, you open your KML file in a text editor and run a **find/replace operation** in any text editor to replace **gx:coord** with **coord.** This will put your file in the right format.

The python script **MapUnique.py** will be extracting the latitude and longitude points from your KML file and writing them to a csv file. A representative sample of the KML file is shown below:

<when>2013-10-05T22:00:23.763-07:00</when>

<gx:coord>-90.3029309 38.6580091 0</gx:coord>

<when>2013-10-05T22:01:17.651-07:00</when>

<gx:coord>-90.3028861 38.6580421 0</gx:coord>

<when>2013-10-05T22:02:07.850-07:00</when>

There is a timestamp, longitude point, and latitude point for each location point you will be plotting. However, **MapUnique.py** doesn’t know how to deal with **gx:coord** and you must preprocess your KML file, by doing a **find/replace operation** in any text editor to replace **gx:coord** with **coord**, your representative sample will have the form:

<when>2013-10-05T22:00:23.763-07:00</when>

<coord>-90.3029309 38.6580091 0</coord>

<when>2013-10-05T22:01:17.651-07:00</when>

<coord>-90.3028861 38.6580421 0</coord>

<when>2013-10-05T22:02:07.850-07:00</when>

**4. Use MapUnique.py to generate a latitude longitude list that can be plotted.**

Once you have preprocessed your KML file you will use **MapUnique.py**.

For this step, you only need to change the line:

**kml\_file = path.join( \ '/disk/path/file.kml')** to the location and name of your KML file and run the python script. This will generate the file **MapPlaces.csv.**

This script has three major functions. First it will open your KML file and extract the longitude and latitude points. Second, it will check each point to see if it is a unique point, and ignore any repeated pairs of points. Third, it will write the unique points to a csv file in latitude/longitude pairs. It turns out that Google Earth plots longitude/latitude pairs, so the python script flips these pairs and writes them in the correct order.

**5.** Write a csv file of any stops you want to display.

**This is an optional step.** If you had any cool stops you want to emphasize, generate a **Stops.csv** file of the form:

Latitude,Longitude,Name as shown in the representative sample:

-79.9961,40.4406,Pittsburgh

-83.7264,42.2708,Ann Arbor

Put this file in the same directory with your python scripts and uncomment the large blocks of commented text starting with:

#Import previously generated Stops CSV File, Write to a new list of lats and longs.

and

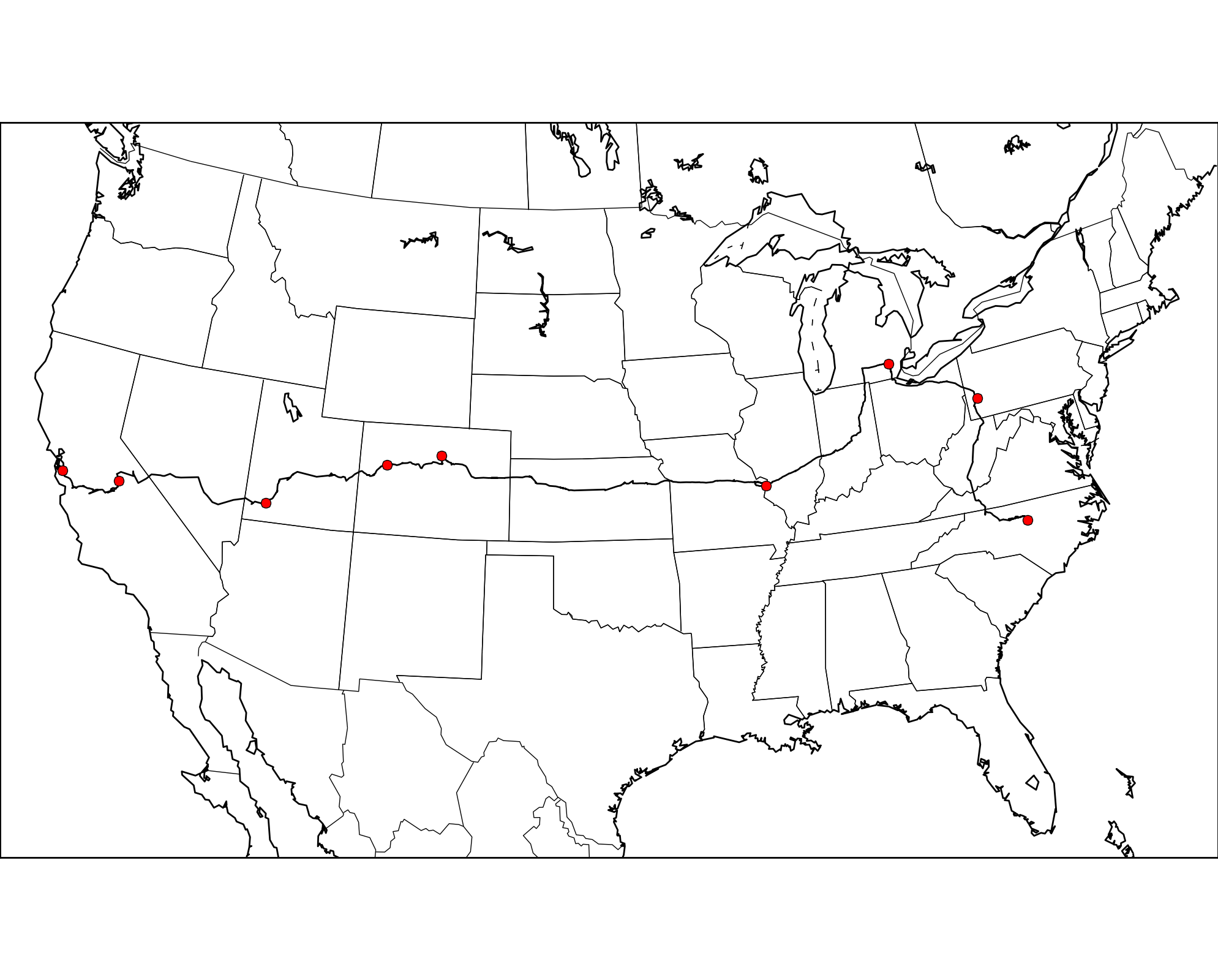
#plot stops lists onto map

**6**. Use ***USMapFromLat.py*** to generate your map in PostScript format.

Once you have a csv file of your location history, you will need to actually generate the PostScript map.

**Run** the python script **USMapFromLat.py**

If you haven’t changed any of the names of the outputs you will get an output **mapUS.ps**. There are lots of manipulations you can make to this map, but they are outside of the scope of these instructions.

You will get a map that looks something like:

The most likely manipulation you will want to make is to the range of the map, which is useful to describe in these directions. At the top of **USMapFromLat.py**  is a block of code:

m = Basemap(llcrnrlon= -121.5,\

llcrnrlat=24, \

urcrnrlon= -65, \

urcrnrlat = 46.5, \

resolution = 'l', \

projection = 'tmerc', \

lon\_0 = -100, \

lat\_0 = 37)

This requests a section of the world map with lower left corner longitude -121.5, latitude 24, and upper right corner at longitude -65, latitude 46.5. If your trip is not across most of the country you will probably want to adjust this range.

**7.** Use **pstoedit** to convert your PostScript map to a dxf file.

You now have a map in PostScript which you can open with **pstoedit** and save to the dxf format.

PostScript is a vector image format but it is not one that 3D CAD packages understand. Unfortunately python doesn’t know how to write to a dxf format, which is an AutoCAD format that most 3D CAD packages understand. Fortunately, we can use a utility such as **pstoedit** to convert between these two formats.

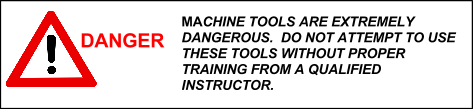
**8.** Import your map in dxf to **your preferred 3D CAD package.**

You should be able to simply open your map in any 3D CAD package, now that it is converted to the dxf format.

**9.** Clean up your dxf and extrude to a solid.

You will have lots of lines that you might or might not want in your dxf file. Delete any that are in places you don’t want lines engraved before you generate a solid.

These extra lines exist because you are asking for a rectangular portion of a large, complex vector map, in addition to the sections and lines of the map that you want you are going to get some you do not want.



**10.** Use **your preferred CAM package** to prepare gcode.

**11.** Machine the map.

These last two steps depend heavily on your experience with, and access to machine tools, and your CAD package. They are outside the scope of this instruction manual. When you are done, however, you will have a map that looks something like:



I used a 6” tall piece of anodized aluminum and used enamel paint on the stops and tracks to make them stand out.