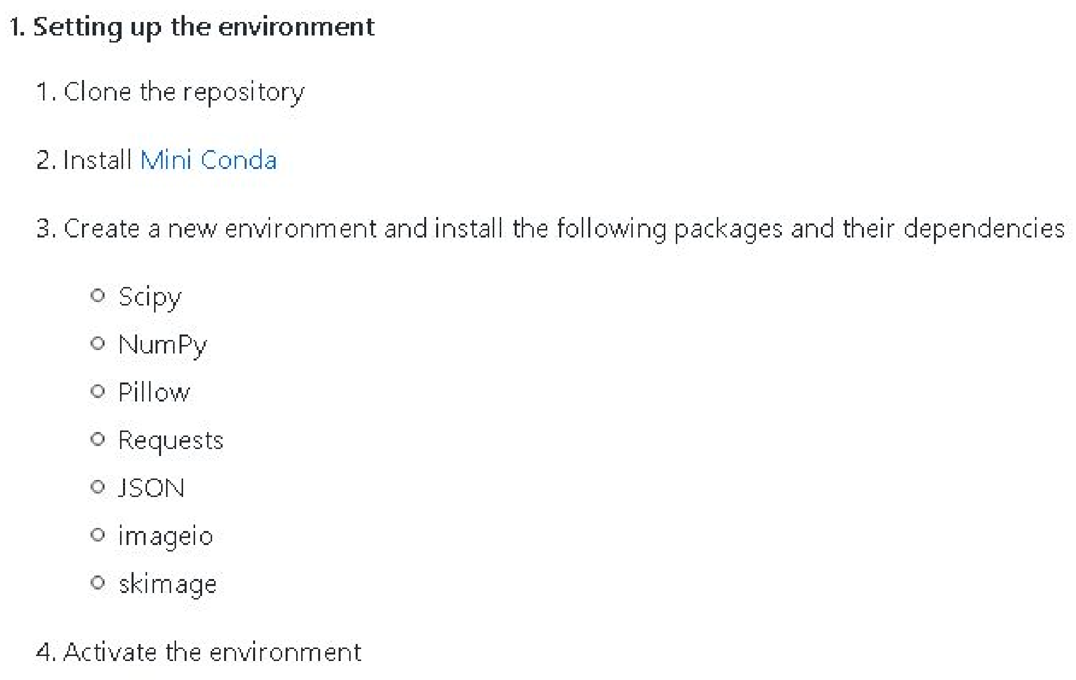
Temporal Distance Map Documentation

Quick Access:
Link to the Heroku based website. https://temporal-distance-map.herokuapp.com/
Be warned that wait times may be rather long(30 min for 6 frame animation creation)

Github Download:
Link to the Github repository. https://github.com/Debusan13/temporal-distance-map
Follow the readme for direct installation instructions.

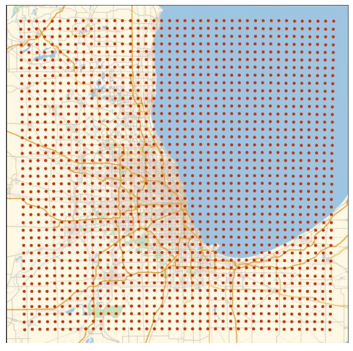


1. Setting up the environment
    1. Clone the repository
    2. Install Mini Conda
    3. Create a new environment and install the following packages and their dependencies
        - Scipy
        - NumPy
        - Pillow
        - Requests
        - JSON
        - imageio
        - skimage
    4. Activate the environment

As well as the Github for the web application.
https://github.com/Debusan13/temporal-distance-map-web

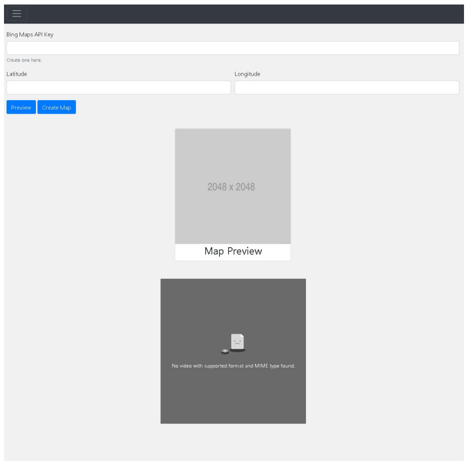
How it works:
We will go into the specifics of what each file in the repository does individually, but we wish to describe the process in broader terms to the benefit of any user or developer. *We left comments in the code that detail what is generally going on*

1. The user enters geographic coordinates for the center of the mesh.



2. The code “overlays” a grid of points that correspond to geographic coordinates
    a. If the user entered 100, -50 as the geographic center in step [1], those points would correspond to (.50000, .50000)
3. Each individual point will be entered into the Bing Maps Distance Matrix that will return the travel time in minutes between the center point and each individual destination
4. The code will then shift the coordinates of the old_x and old_y to a new set of coordinates based on their travel time (using distance formula and geometry)
5. The transformation of the old (x,y) values to the new will be accompanied by an alteration of the image by shifting the colors associated with each coordinate and interpolating the points that lack data.

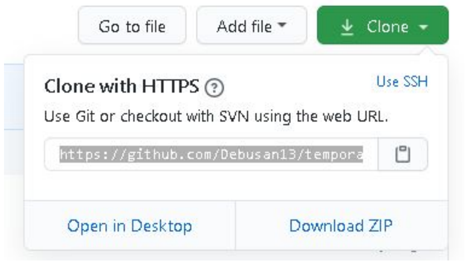
Utilizing Web Application:



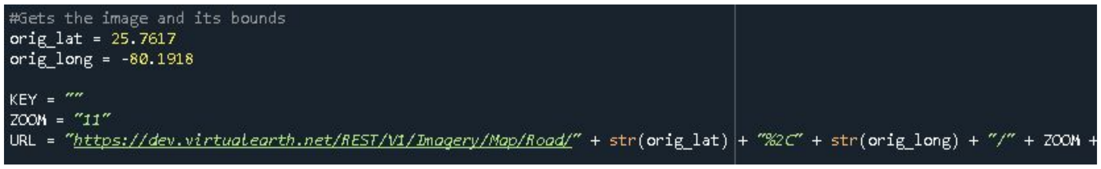
1. Access the link. https://temporal-distance-map.herokuapp.com/
2. You will be met with the interface pictured above. You will be asked to enter a Bing Maps API key as well as a set of geographic coordinates.
3. If you do not have access to a Bing Maps key, you will need to get one (for free) from the website linked below the “Bing Maps API key” prompt.
4. Otherwise, simply enter the required information and wait roughly 30 minutes.
5. After it is finished processing, a 6 frame animation will appear depicting the transformation of the location.

Running the Code:

1. After cloning the repository by either...
    a. Typing git clone https://github.com/Debusan13/temporal-distance-map.git into a terminal
    b. Downloading the zip file https://github.com/Debusan13/temporal-distance-map (under the clone tab)


Go to file
Add file
Clone
Clone with HTTPS
Use SSH
Use Git or checkout with SVN using the web URL.
https://github.com/Debusan13/tempora
Open in Desktop
Download ZIP

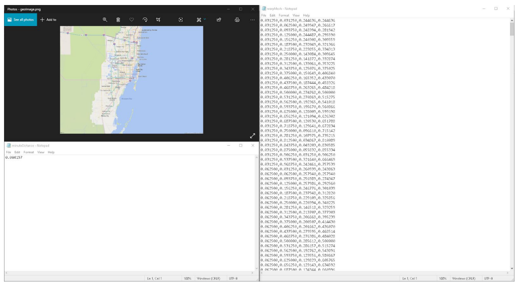
2. Open the file titled “MapLogic.py”.
3. On lines 14 and 15, change the variables orig_lat and orig_long to match the geographic coordinates of the location you wish to view.
4. On line 17, enter the Bing Map API key that you will be using
    a. If you have no Bing Map API key, a new one can be acquired here. https://www.bingmapsportal.com/



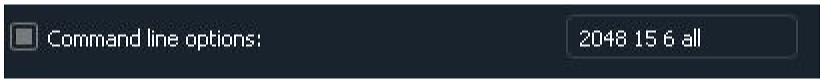
```
#Gets the image and its bounds
orig_lat = 25.7617
orig_long = -80.1918

KEY = ""
ZOOM = "11"
URL = "https://dev.virtualearth.net/REST/V1/Imagery/Map/Road/" + str(orig_lat) + "%2C" + str(orig_long) + "/" + ZOOM +
```

5. Now simply execute the code, this file will write a static (unwarped) image of the location to “geoImage.png”, a file called “minuteDistance.txt” that indicates the euclidean distance of one minute on the map, and a large file “warpMesh.txt” that holds the mesh of points in the format “old_x, old_y, new_x, new_y”

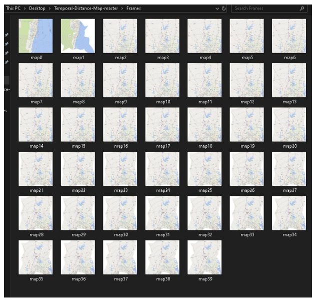


6. Now close “MapLogic.py” and open “warpAnimation.py”
7. Run the code with Arg values representing
    a. [1] Map Resolution
    b. [2] Distance between rings in minutes
    c. [3] Amount of frames to render
    d. [4] Distinguishment of which frames to render



Command line options: 2048 15 6 all

8. Typically, we ran “warpAnimation.py 2048 15 6 all” but there is versatility to fit user needs
9. “warpAnimation.py” will have the lengthiest run time but it also has a progress bar and gives an estimate of time remaining. The file will output individual frames into the folder “Frames” that will later be used to form the animation of the locations warping. *Image showcase a Frames folder with two different warpings present - difference in color*



10. Once “warpAnimation.py” finishes, open “makeAnimation.py”
11. Run this code and it will give you the “warpAnimation.mp4” which will showcase the warping of whichever location you chose.

Issues During Development with Code

1. Accessing the Wolfram Alpha code as it was locked behind a paywall and we only had a limited time with it.
2. The Bing Maps API does not allow the gathering of time data if there are more than 10 calls being made at a time, for reference out code makes slightly under one thousand calls.
3. Getting into contact with the old code authors

Issues During Development with Web Application

1. Serving a newly written static file via Flask, which was resolved by using a placeholder image.
2. Serving video as both a downloadable element and in an inline player.
3. The code ran for so long that the web interpreter would treat it as a run time error and abort.
4. Multiprocessing and using dynos on Heroku in order to show users how long the application will take to run
5. Using a redis server framework to pass jobs to the multiprocessing units.
6. Allowing Heroku to accept the redis server.

Possible Further Development:

1. There are commented out sections of the code that need further development before they are implemented
    a. In “warpAnimation.py” there is a section called “imPoints” is commented out
        i. Typically this section would overlay the names of cities on the map but the route to getting the coordinates of nearby cities in Bing Maps is rather ambiguous and will need additional effort.
    b. In “makeAnimation.py” there is a sedition of commented code that involves color
        i. This code works and will remove all the city names and highway signs that blur a lot of the map. We planned on implementing this code if “imPoints” is ever developed properly.
2. Shortening the overall runtime of warpAnimation
3. Making less calls to the Bing Maps API
    a. Each key can only make a certain amount of calls, at the moment our code uses 962 calls to the API each time it is run.

Folder of Relevant Visuals:
https://drive.google.com/drive/folders/1mUdzWT4cqynNE_K6SRBm3O0MsGCSsraM?usp=sharing