Elevation-based Workout Navigator

Eric Atwood, Steve Borst, Yifu Liu, Jeff Muri, Kailash Nathan Srinivasan

Github: https://github.com/CS520-Elevate/Elevate

Description

Elevate is a street navigation application built around workout optimization for Amherst,MA. Given starting and ending travel points, it will display a route based on a level of workout intensity specified by the user. Elevate will calculate a route with greater elevation changes for more intense workouts, and smaller elevation changes for less intense workouts.

Features:

- Desktop interface to input GPS coordinates for starting and ending points
- Options for easy, medium, and hard mode
  - Easy produces the shortest path between each point
  - Medium produces a path with increased elevation change between points with a constraint.
  - Hard produces the path with the most elevation change
- After requesting navigation, a map appears with the desired path
  - Can zoom in/out, drag to see around the map

Architecture

- Elevate uses a model-view-controller architecture.
- The view includes a user input field implemented with the Tkinter framework and a map displayed as an HTML webpage.
- The model is a simple Python class that tracks user input data.
- The controller is mainly based on the osmnx and networkx libraries in Python.
- Folium for the data wrangling and visualisation of maps.

Algorithm

Easy mode uses OSMNX version of Dijkstra algorithm with distance as the weight measure to minimise distance between the nodes.

Medium mode uses a modified version of Dijkstra with constraints between the length of the route and the elevation while limiting the total distance between the two locations to 1% of the shortest path.

Hard mode uses OSMNX version of Dijkstra algorithm with Impedance as the weight measure to maximise elevation between the nodes while limiting the total distance between the two locations to 30% of the shortest path.

Observations

- There seems to be little difference between medium and hard modes over shorter distances.
- Most of the times, the algorithm was able to find the path that was same as the one with shortest distance but with minimum or maximum elevation.
- Doing an extensive search for all possible paths from origin to destination without a constraint for the extra distance affected the performance in hard mode.
- Obtaining minimum elevation gain was easy due to Dijkstra algorithm.
- Adding a distance constraint between the shortest path and total distance proved to be harder as the path with the required elevation might not be ideal because of the constraint.
- The model seemed to perform better on distances over 4000m over different modes as compared to smaller distances as the constraint mostly failed.

Screenshots

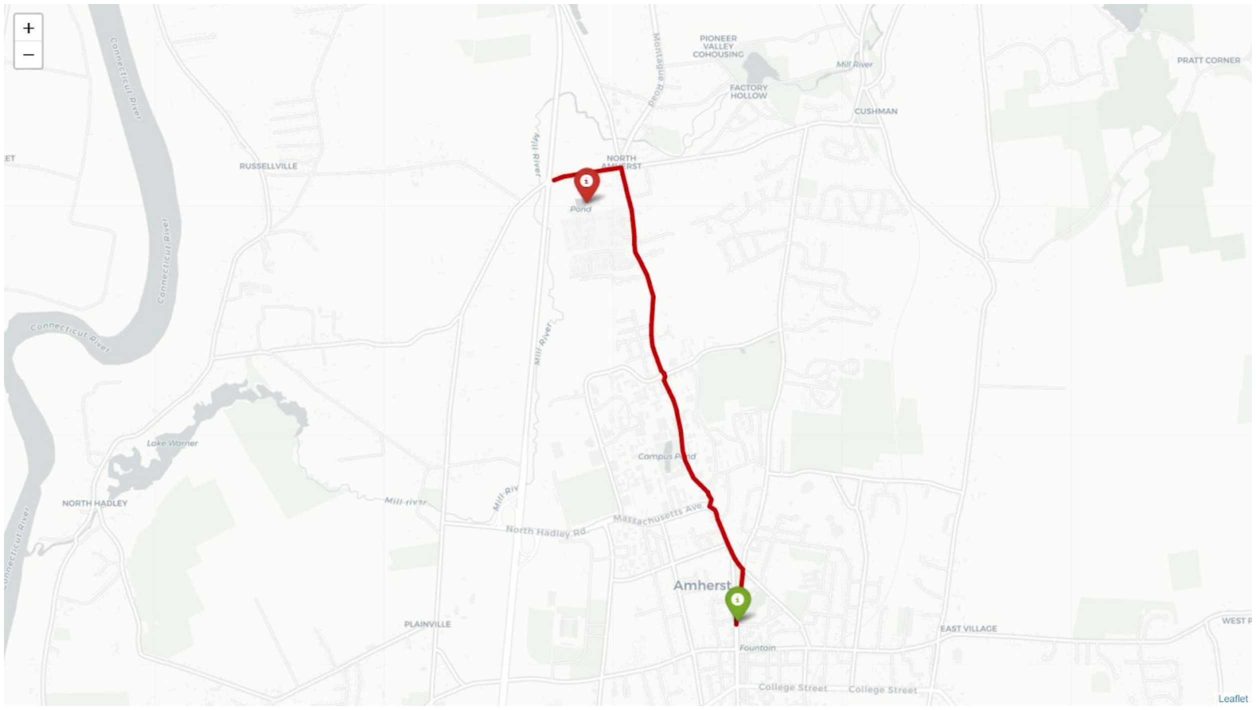
Green marker denotes the starting location and the red marker denotes the destination.

Between the same source and destination , these are how the easy and hard mode routed between the nodes.

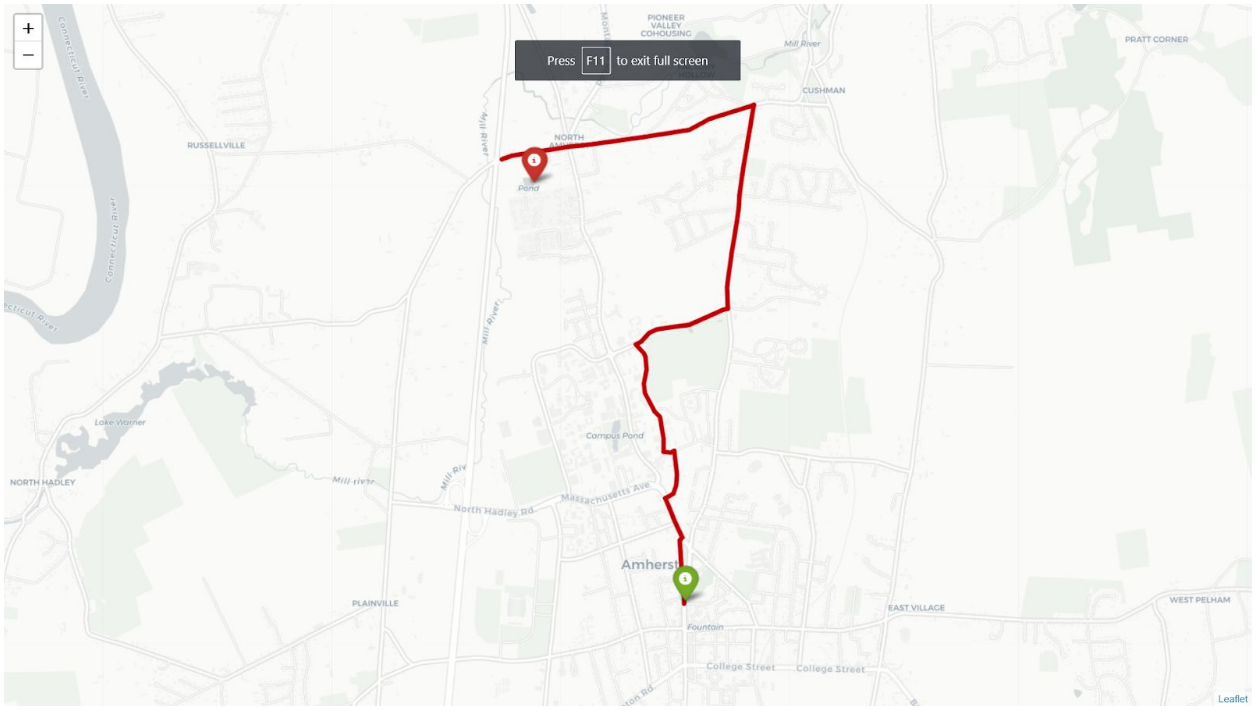
First example was the navigation from Amherst Post office to Brandywine apartments in Amherst over easy and hard modes.

Second example was a shorter route from Haigis Mall to Downtown Amherst over the easy and hard m

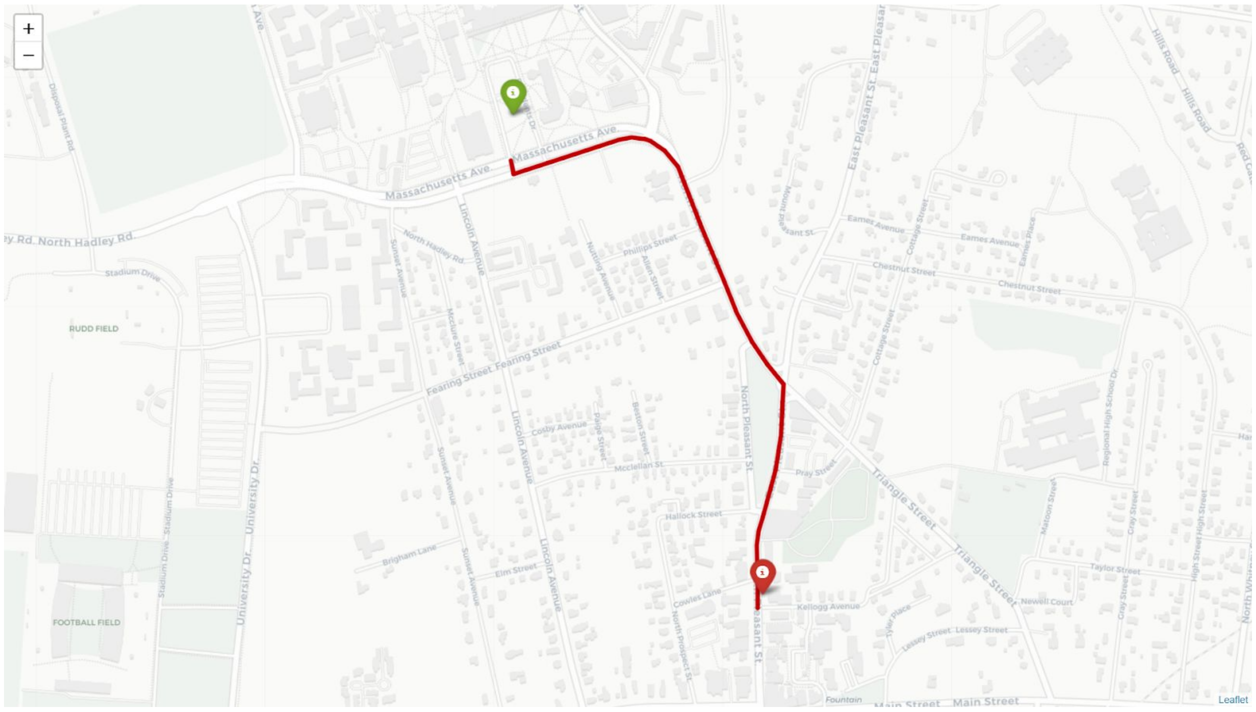
Amherst Post office to Brandywine apartments



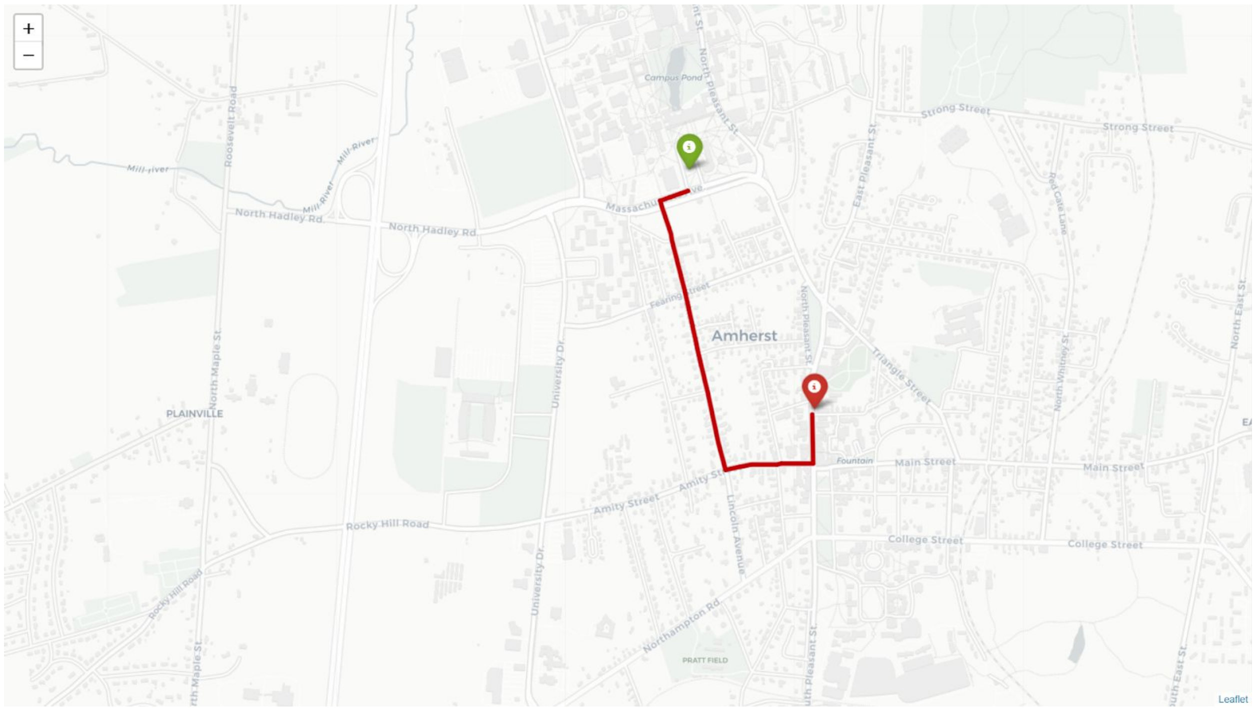
Hard:



Haigis Mall to Downtown Amherst

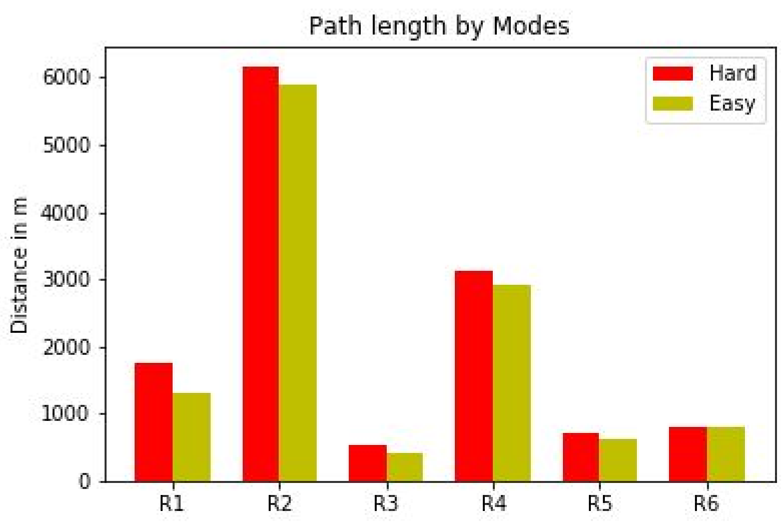


Hard:

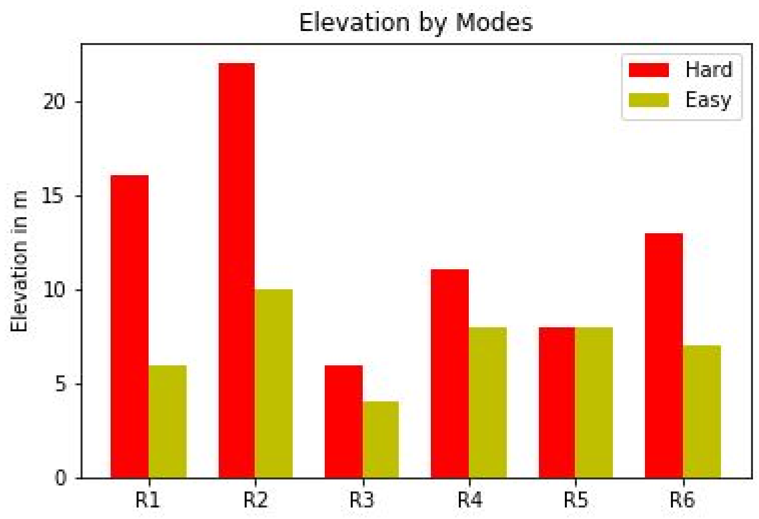


Performance:

This is how our model performed between 6 routes over varying distance in terms of distances of the path.



In terms of elevation


Elevation by Modes
Hard
Easy
Elevation in m
R1
R2
R3
R4
R5
R6