Algorithmic Map Recognition and Edge Detection with Point to Point Pathfinding

Computer Science NEA

Name: Rubens Pirie

Candidate Number: 1749
Centre Number: 58231

Centre Name: Barton Peveril Sixth Form College

Contents

1 Analysis

1.1 Statement Of Problem

Maps, as you would think of them today, have been around since 6th century BC and since then have been in constant use by people in their day to day lives. The more modern version of maps, for example Google maps or Bing maps have only been around since the late 1990's. The problem that I am going to be solving is map path finding. Currently not all roads and paths are logged and entered into a searchable format. The only way some people have to navigate terrain is through the use of old style paper maps. The problem with paper maps is that they are not easily, at a glance, used to find a path from point to point. As well as this sometimes are not easy to comprehend just by looking at them with various terrain features.

(a) Map without labels on roads

(b) Map with labels on roads

Examples of maps with and without labels taken from Google Maps©

This can cause issues for people who live out in areas which have not been mapped. This is because they cannot create easy to follow routes with the click of a button. Therefor, causing people who live in rural areas to waste time getting used to the routes they have to take to go anywhere. Overall, the problem I am going to be creating a solution for is how people are unable to easily go from point to point at the click of a button and be easily able to, at a glance, interpret the map without prior experience.

1.2 Background

When people usually want to go about planning a journey they will use a service, for example Google Maps to get from one location to another. This usually takes the form of clicking a location and then selecting an origin. This isn't always possible however, this can be for a multitude of reasons it seems however I will briefly go over some below:

1. **Either the destination or origin location(s) are not in the service's database.**
2. **The destination and origin have no clear defined path between them.**
3. **Either the destination or origin are off any predefined track.**
4. **The travel method the user has selected is not able to traverse the terrain between the origin and destination.**

Some of these I believe are out of the scope of this project however once the interview has been conducted with the end user I will have a better idea of the needs that my program needs to for-fill.

Finally,

1.3 End User

1.3.1 First Interview

In order to get a better feel for the objectives and functions that my program should complete I interviewed with an end user, Mrs Mandy T. I believed that she was an appropriate candidate for this project due to the fact that she has to drive into work every morning. Along her route she has to deal with Google Maps which do not cover all of the roads in her area. Therefor in the following questions I asked her some questions gauge her priorities when it comes to web mapping.

1. **When using web maps (e.g. Google Maps©) what are the key features you look for?**

   "A scale! WHY is it lost so often when Google Maps is embedded?! Then it depends what type of map I'm looking at... if it's a road map then....roads! Size/type of road is important and things like one-way restrictions. If it's for e.g. walking...footpaths/bridleways and parking are important. "

2. **Have you ever experienced a faulty or mislabeled part of an web map or has said map ever been inaccurate?**

   "Yes"

3. **Do you often use web maps in your day to day life, if so how?**

   "Yes, **NEEDS TO BE ADDED TO**"

4. **In your opinion do you feel that web maps are vital to every day life if so why or why not?**

   "No. I passed my driving test before we had sat-nav or internet, so clearly they're not vital - we survived without them!
   They are quite helpful though as we used to have to buy a new road map every year, whereas web maps can be updated as things change, instead of only annually!"

5. **What makes a good user interface for a web map?**

   "Clarity and simplicity. Nothing needlessly complicated."

6. **How do you use web maps (e.g. long journeys, short journeys, school runs)?**

   "Route functionality on long or unfamiliar journeys. Using them a lot at the moment as am planning a holiday overseas. The maps are useful to see whether accommodation and restaurants will be walking distance, and what options there are in each location etc."

7. **Do you feel a tutorial would be beneficial to aid in the use of the map or should the focus more be spent on intuitive ease of use?**

   "If they're easy to use, a tutorial would be surplus to requirements, so ease of use is more important. "

8. **Would it be beneficial to store old routes?**

   "Not really (is this a routing question?). I don't know what purpose that would give, unless I was being accused of something and needed to use the route as evidence of being in a certain location! It could be use full in the context of frequently traveled routes however if this was the case I would know the route by heart anyway."

9. **What forms of transport should the map include?**

   "(I think this is a routing question not a map question) Walking, bike/horse, car, bus, plane, ferry. If just a map question, then the map should include footpaths, bridle paths, roads, ferry routes"

10. **If there was one feature you could have implemented in an existing solution what would it be?**

    "To be able to post a question about a specific area and have a person who is local to that area answer it."

1.3.2 Evaluation of First Interview

Overall I feel that this interview gave me valuable insight into the requirements of my end user. As well as this my end user made it clear to me that there are two overriding parts of this solution. The map recognition aspect of it and the path finding aspect. Going deeper into the path finding part of this project I will need to do research on the different methods that will be used to achieve this and some of the possible data structures I could use.

1.4 Initial Research

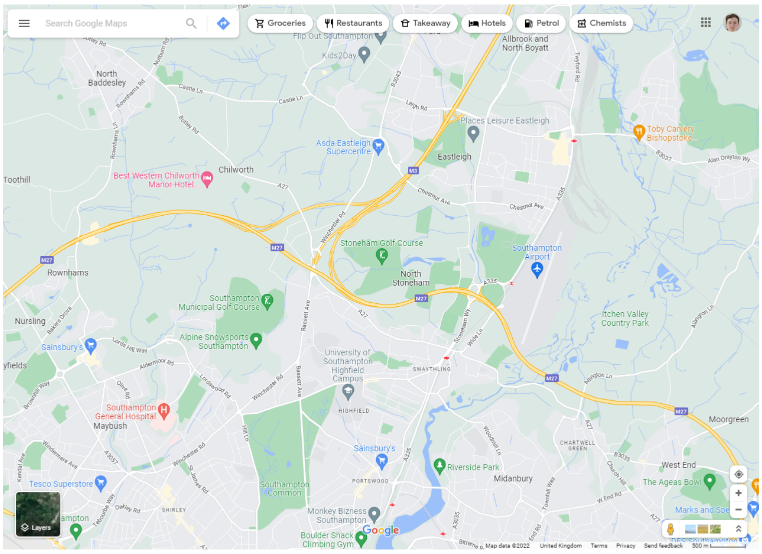
1.4.1 Existing Solutions

Below each overview passage I have included an image of each map for comparison of their GUI's. These will be used as inspiration as to how my final solution will look as well as serving as examples of how the GUI can sometimes become overly complicated. This is especially the case with Bing Maps as when you initially access it you are flooded with popups and extra options.

Google Maps

As aforementioned this is one of the most used forms of interactive web mapping in use at the moment. It has been in use since 8^{th} February 2005. As it exists now it is an interactive world map with routing features built in. It provides detailed information about geographical places and regions around the world. Unlike some of its competitors it also offers aerial and satellite images of places around the world aiding in navigation of terrain.

As well as its map viewing capabilities it also offers partial route planning and live route tracking for cars, bikes, walkers and public transport. It provides instantaneous and real time feedback while you are moving however the one big caveat to this is the fact that it will require an internet connection to run, something that is not always available.

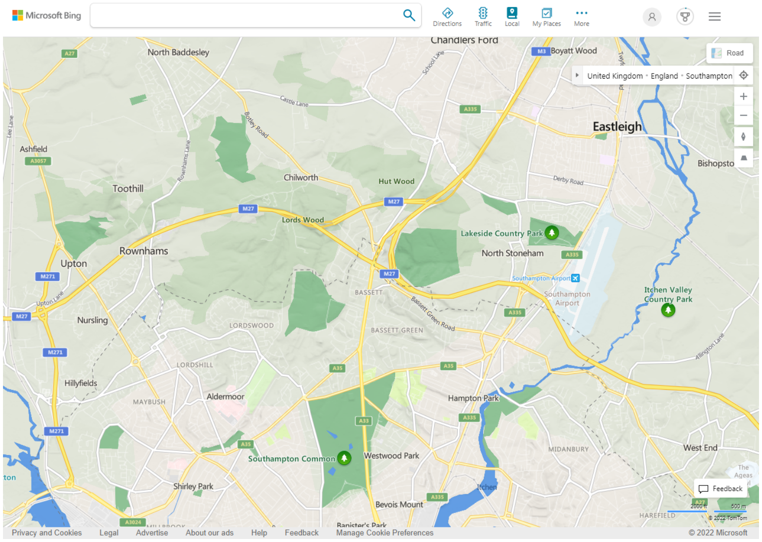


(a) Example of Google Maps' GUI

Sourced from Google Maps©

Bing Maps
This is another form of interactive web mapping. This is a more plain version of Google Maps at first glace. This is due to the fact that it does not have as many features as Google Maps. This does have its advantages due to the UI seeming less cluttered and more accessible. Similar to the Google Maps it also offers route planning and map traversal as well as live traffic updating. Bing maps unlike Google Maps boasts a more open API and easier programmatic interface for developers to be able to interface with their program.

Bing maps also still includes the feature which allows users to create their own maps based on their own data. Unlike google which did have this feature until they discontinued it. I believe that this could be something that would be beneficial to my program, allowing people to take a photo of their own map and have my solution compute it into a routable map.

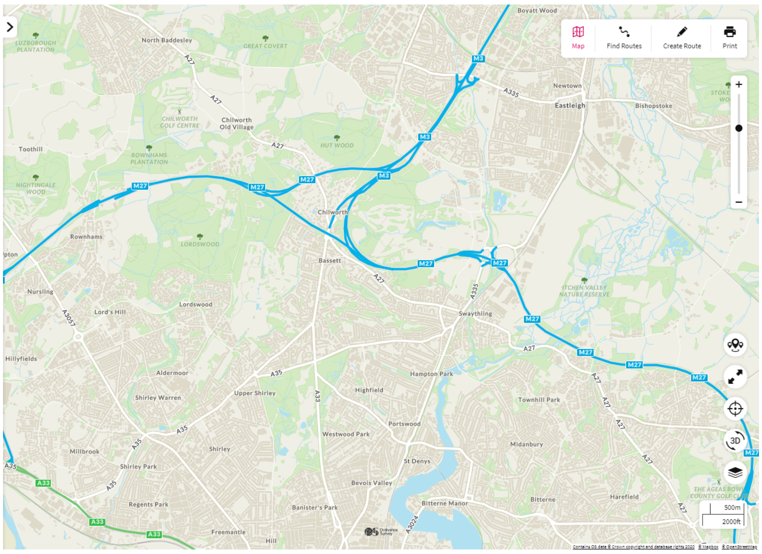


(b) Example of Bing Maps' GUI

Sourced from Bing Maps©

OS Maps
This is a different take in web mapping compared to Bing and Google Maps. With Ordnance Survey their focus was on the accuracy of their maps hence they do not have as an extensive routing system. If you wanted to go from point to point on an OS map you would have to plot it by hand. However if you wanted to go on an exercise trail on the other hand they are very well suited for this and as such have an extensive list of pre-planned routes.

Similar to Google Maps, and in a limited capacity, Bing maps; OS Maps allow you to view their maps in different forms such as 3D and topographic however in order to access these you will have to access their premium plan therefor for the average user this is not a viable option and a hindrance. It is good to note however that the other variations on the map of the UK, and this holds true for all of the aforementioned maps, that the satellite view and other views are not necessary and could in fact be a hindrance.



(c) Example of Ordnance Survay's Map GUI

Sourced from OSMaps.com©

Existing Solutions Conclusion
In conclusion, I have found that the existing solutions that are available are all very well designed and well implemented. I have found that they are easy to use and rather intuitive however, for the average user who just needs to get from A to B in the most economic way possible they are overly complicated. As well as this I have found that with the exception of OS maps both of the other solutions require an internet connection to get the best use out of their maps, this is something which I believe I should avoid. This will mean that all calculations will have to occur self contained within the program, not allowing the use of external API's.

1.4.2 Possible Algorithmic Solutions

There are, as aforementioned many existing solutions which work in various ways, in order to make my solution unique and functioning I am going to have to incorporate many different algorithms and theories.

Edge Detection
First of all I will need some way of recognising a map and parsing it in some way. The way that first springs to mind is edge detection. This is a way of taking an image and computing where there are changes in contrast or brightness which could be considered an edge. There are many forms of edge detection out there all of which work in various ways, the main things they look for however are discontinuities in depth, discontinuities in surface orientation, changes in material properties and variations in scene illumination. All of these factors combine and allow a program to decide if there is an edge in an image.

A simple edge detection model can be extremely effected by natural blur or artifacts in an image. In order to mitigate this there are smoothing algorithms that can be used to blur and smooth edges causing the impact of artifacts to be avoided. The common term when referring to artifacts and erroneous data in an image is *noise*. I believe it will be beneficial to include some of these in my solution, this will be something to look into in the **Further Research** section.

Taking a quick look at one form of edge detection, Canny Edge detection, it is relatively simple in its implementation. It has only 5 steps, first removing noise with a Gaussian filter then applying bounding to the image and finally performing hysteresis threshold. This is the most common form of edge detection that I have come across in my research however there are others. A rather different example of edge detection is Kovalevsky edge detection. Unlike canny edge detection this does not care about the luminosity of the image and goes based of the colour intensity in each of the channels.

Graph Forming

This is not so much a possible algorithmic solution but more of something that my solution will have to achieve. Once the image of the map has been altered and the edge detection has been performed, I will be left with an image which has white lines where there "edges". From this I will need to create a weighted graph as well as an unweighted graph.

During my research I have failed to come across an existing solution to this problem. As well as this looking through some examples that people have uploaded it seems that sometimes the edge detection does not yield a fully connected image. This could prove to be an issue as it would add the possibility of isolating certain roads.

I feel that I need to look more into this and come up with my own solution during the prototyping stage, and come up with my own algorithmic way of generating it.

1.4.3 Key Components Required

After doing my initial research and a brief look at the existing solutions I have come up with, what I feel, is the main 4 Components that I will need to build my solution.

The Graphical User Interface

Talking to my end user made it clear to me that in order for the program to be usable by the wider population it would need to be clean and uncluttered. This leaves me in a difficult position due to there being a limited amount of frameworks that are available to me. I have two sets of possibilities:

1. A Local App Run on Device
2. A Web Based Application

Each of these have their advantages, if I where to go with a locally run app I could make it in the console keeping it simplistic and easy to use. However if I do use the console it would limit this solution to a computer which could be seen as going against the idea of this problem. On the other hand, if I where to go with a web server based application this would yield much better compatibility with all devices since all you would need is access to a web browser. This, by its very nature, means that you would need an internet connection which is also a problem which I was hoping to fix.

The solution then I believe is to make it both a locally based program with the option for it to run a web server. However I will need to specify one over the other to begin with to make sure that the program is working either way.

Regardless of which one I choose I will conduct some form of testing where I will allow, through a survey, people to specify what makes an easy to use and intuitive.

Image Manipulation and Edge Detection
This is perhaps the most important part of the project since without this I would not be able to continue to path find the image of the map. Looking at my research I feel that there will be a combination of

Graph Creation and Representation
From lessons which we have had in class I have been shown that there are 2 reasonable ways of representing a graph in code, this includes an adjacency matrix and an adjacency list. Both have their advantages and disadvantages. An adjacency matrix is good when you have a reasonably connected graph which has weights, this is due to it being easy to access and traverse. As soon as you have a sparse graph however it becomes very memory intensive which is unnecessary considering that there will be very few of the cells with actual data in them. This is when the adjacency list comes into play, the reason that I am reluctant to use this form of representing a graph is that when performing some of the various graph traversal algorithms it can incredibly difficult and pointless to adapt them when by adapting them you effectively generate the adjacency matrix.

Graph Traversal and Output

1.5 Further Research

1.5.1 Dive into Specific Algorithms

After doing some research it seems that there needs to be a set of definitions before I go any further to avoid confusion. This is because during my time on Wikipedia there are sections where several terms are used interchangeable where I feel they are not the same. Each of these definitions are as defined by me and are not necessarily the official definitions since they do not explicitly exist. They are as follows:

1. Graph Traversal: The act of routing or searching through a graph from one node to another, either using an algorithm or by another means.
2. Graph Routing: Graph traversal in a *weighted undirected* graph.
3. Graph Searching: Graph traversal in a *unweighted undirected* graph.
4. Graph MST: The minimum spanning tree of a graph which must be weighted.

The difference is slight however the key takeaway from this is that when I am referring to a Routing algorithm I am referring to one which works on a weighted graph. And vice versa if I am talking about a searching algorithm this is referring to graph traversal on an unweighted graph.

Black and White Filter
In order to allow the program to function, assuming that the canny edge detection was chosen we do not need the colour data of the image. In order to remove this a filter is used, this one is the industry standard since it takes into account how prevalent red, green and blue are rather than taking an average which could become non representing of the real case.

$$\beta = 0.299 * \alpha_b + 0.587 * \alpha_g + 0.114 * \alpha_b \begin{cases} 255 & \beta > 255 \\ 0 & \beta < 0 \\ \beta & \beta \in [0, 255] \end{cases}$$

If an averaging was used it would just be, this is also known colloquially as the "quick and dirty" method.

$$\beta = \frac{(\alpha_b + \alpha_g + \alpha_b)}{3}$$

Gaussian Filter
This is the first step of 5 in terms of performing Canny Edge Detection. Applying the Gaussian filter to the image will smooth out the image and remove any noise. It does this by taking a section of the image, sometimes referred to as a kernel and performing an equation on it. Once it has computed the equation it sets all of the pixels inside the kernel to this value. The following is true for a kernel size of $(2k+1)*(2k+1)$. It takes two changeable parameters σ which denotes the amount of blur to apply and k is the kernel size. As well as being one of the key steps in canny edge detection it is also a vital component to most edge detection programs since noise can caise errors in the final image.

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(- \frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2} \right); 1 \leq i, j \leq (2k+1)$$

Since the Gaussian kernel I would be using would always be centered around the origin $(0, 0)$ I can use a simplified version of the Gaussian distribution equation. This is as follows:

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp \frac{-(x^2+y^2)}{\sigma^2}$$

I can afford to remove the $(i - (k+1))$ section due to the fact that I am not having to calculate the Gaussian distribution at a non-centered location. One notable thing to mention is that in many cases it is not necessary to calculate the Gaussian kernel by hand and an approximation can be used. The example below is the approximation when σ has a value of 1.

$$B = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * A$$

1.5.2 Second Interview

Now that I have done some more research into the various ways there are to complete this task I have formed some more questions to ask my end user to get a solid and defined list of objectives for the program. AI will couple this with my research to form a complete plan to form said objectives. As well as this however the second interview will allow me to correct any inaccurate questions that where asked in the initial interview. This is because after I received my initial responses I realised that I needed to be more clear with what I was asking and the information that I wanted back.

1. **Bobbert?**

   bobbert.

2. **Cobbert?**

   cobbert.

3. **Dobbert?**

   dobbert.

4. **Fobbert?**

   Fobbert.

5. **Norbert?**

   norbert

6. **Dilbert?**

   dilbert.

7. **Bobbert?**

   bobbert.

1.5.3 Evaluation of Second Interview

After conducting this second interview I feel I now have a firm understanding of what I need to achieve with this program. I will also take this opportunity to create a prototype of the the different parts of the program to gauge the difficulty of the program and any problems I may encounter before moving onto the final solution.

Apart from that however I feel the interview went…

1.6 Prototyping

1.6.1 Prototype Objectives

Before I begin the creation of my prototypes I will create a list of sections I wish to complete by the end. This will allow me to keep perspective and make sure that the prototype remains on track. I have decided that the parts of my final solution are:

- A version of edge detection
- A graph class with basic traversal
- A forms interface for showing images

1.6.2 Edge Detection

For the example of edge detection which I am going to prototype I have chosen Canny Edge Detection, this is the most common of the types of edge detection and is relitivly simple.

1.7 Objectives

After conducting the initial and second interviews and reflecting upon the results of my research I have formed a list of objectives that the program must meet to be considered complete. As well as the base objectives I have also, with help from my end user, come up with extensions which will increase the effectiveness of my solution overall.

1. The Program must have way to input a Map
   - 1.1 The Program should be able to parse a map from a file, including
     - 1.1.1 A photograph of an map
     - 1.1.2 A screenshot of an existing map
     - 1.1.3 A hand drawing of suitable quality
   - 1.2 When the user inputs a map, the program will ask them
     - 1.2.1 What type of map they are inputting
     - 1.2.2 How they would like the result of the map
     - 1.2.3 How harsh they would like the edge detection to be
     - 1.2.4 The resolution of which to parse the map too
     - 1.2.5 Whether they would like to path find through the map or form a Graph MST
       These are just some examples of prompts
   - 1.3 The inputted map should be converted into a graph
   - 1.4 The map (in graph form) should be able to be traversed
   - 1.5 If any error occurs during the map input process an appropriate error should be displayed and the program should continue to run
2. The Program must allow Map Traversal
   - 2.1 There should be Multiple Traversal Algorithms Available to be chosen from.
     - 2.1.1 The Program should implement Routing Algorithms
       - 2.1.1.1 This includes Dijkstra's algorithm

2.1.1.2 This includes A*
2.1.1.3 This includes Shortest Path Faster Algorithm (SPFA)
2.1.2 The Program should Implement Searching Algorithms
2.1.2.1 This includes BFS (Breadth-first search)
2.1.2.2 This includes DFS (Depth-first search)
2.1.2.3 This includes Greedy Best-First Search
2.1.3 The program should implement a Minimum spanning tree
2.1.3.1 It should implement Primm's algorithm
2.1.3.2 It should implement Kruskal's algorithm

2.2 Depending on the option that the user chooses they can either
2.2.1 Decide a specific algorithm to use
2.2.2 Have the program compute all and use the most efficient one
2.2.3 Have the program compute all and let the user select one

3. The Program must have a Clear and Simplistic GUI
   3.1 At a glance the user should be able to ascertain which step they are at in the process.
4. The Program must have image processing capabilities
   4.1 The program should be able to perform edge Detection

Extension Objectives

5. The program should be able to output
   5.1 The map in a binary file format
   5.2 The routed map with path drawn once
   5.3 Various stages of image Manipulation
6. If the user chooses the program should be able to run a web server
   6.1 The web version should have a similar speed to the local version
   6.2 The web version should have a similar feature set to the local version
   This only applies to features that are possible to work in the web, they may be displayed differently that the local application due to different frameworks
   6.3 The web interface should be easy to use and intuitive

1.8 Modeling

2 Technical Design

3 Program Testing

4 Evaluation