Generating GPS Art

James Harvey
Department of Computer Science
University of Warwick

Supervised by Ramanujan Sridharan

2 May 2023

WARWICK
THE UNIVERSITY OF WARWICK

Abstract

GPS art is the creation of visual compositions by mapping paths using a GPS tracking device. GPS artwork has gained popularity recently as a unique form of artistic expression that combines technology, physical activity, and creative design. However, plotting and designing these routes is an arduous process due to the restrictions imposed by street networks and urban landscapes. With no software or approaches currently providing a solution to this issue, this paper explores how an algorithmic approach can automatically generate GPS drawings, making the procedure more accessible and encouraging exercise.

Keywords: GPS Art, Routing, OpenStreetMap, Networks, Physical Activity,

Acknowledgements

My sincere appreciation goes out to Ramanujan Sridharan, who served as my supervisor and provided me with invaluable guidance and support throughout the course of this project. I would like to express my gratitude to Geoff Boeing for creating and maintaining the OSMnx library, which was used extensively in this dissertation. Finally, I am indebted to my friends and family for their constant encouragement, love, and support during my undergraduate studies.

Contents

List of Figures

1 Introduction

GPS Art, also referred to as GPS Drawing, is a technique in which an artist utilises a Global Positioning System (GPS) device to follow a predetermined path to create a visually striking image or pattern on a large scale [1]. The GPS device can be seen as the "pen" and the surrounding landscape as the "paper" [2]. Artists normally run or cycle the route-while cars, boats, and aeroplanes are utilised to create larger pieces. In 2010 one artist, Yassan, even quit his full-time job to embark on a six-month-long journey across Japan to spell out the words 'Marry Me' [3][4]. After racking up 7163 kilometres on his bike, his girlfriend, Natsuki, gladly accepted his proposal. He also received a Guinness World Record for the largest GPS drawing, which still stands to this day [5]. The final route can be seen in figure 2. The number of people participating in the creation of GPS art is increasing as GPS-capable devices are becoming more common and built into everyday devices like phones. The unique blend of creativity and fitness is motivating runners and cyclists to change up their routes, whilst others enjoy the creative challenge of drawing whatever comes into their minds.

There are three main approaches to creating GPS art depending on the desired mode of transport and the landscape around them. The first of the approaches is where the artist is restricted to only the roads, trails and paths around them. The next approach is a freehand GPS drawing. This can be performed in large public areas, open air, or the sea. The only restrictions are the bounds of the open area; therefore, the final art can be more detailed. These open areas can include parks, beaches, and fields for artists cycling or running the route. By default, artwork created by planes and boats is freehand as no geographical restrictions exist. The last approach uses a connecting-the-dots method to create waypoints that make up the artwork by turning the GPS device on and off at specific points along the drawing. This method bypasses the limitations imposed by a street network, as the lines between waypoints do not have to be physically traversed by the artist. In this project, the focus is the former approach as the latter approaches have semi-trivial solutions, whereas the constraints caused by the street networks prove tricky to design artworks where roads, trails and paths have to be followed.

Typically, the shape of the design is influenced by the layout of the road topology, where artists identify distinctive characteristics within the road network and integrate them into a larger image. Sometimes artists have preconceived designs in mind and must determine the size and orientation of their route to ensure a similar correspondence. Nonetheless, accomplishing this task is challenging since there are numerous routes to consider, and artists frequently need to compromise the accuracy of their image since there may not be any routes that are an exact match. Furthermore, once an artist has identified a satisfactory route, it is difficult to determine if it is the optimal match within a particular radius of their intended radius. Given the lack of currently available software, creating a tool that can generate a route closely resembling a desired artwork would be a useful tool.

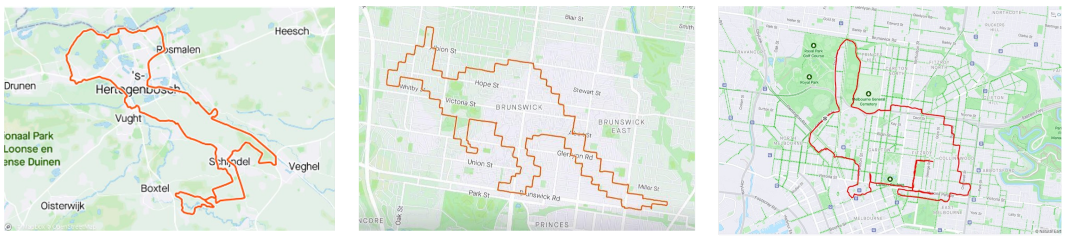


Figure 1: Examples of GPS art

An example of GPS art restricted by trails, roads and paths can be seen in figure 1. The drawings represent the shapes of Italy, a kangaroo, and a bunny rabbit, respectively, from left to right. Figure 3 shows an artwork showcasing both the restrictiveness of having to stick to a street network as well as the freedom of open areas. Designed by Jason Wood, Traverse Me has been drawn on foot by walking the entirety of the University of Warwick and surrounding areas [6].

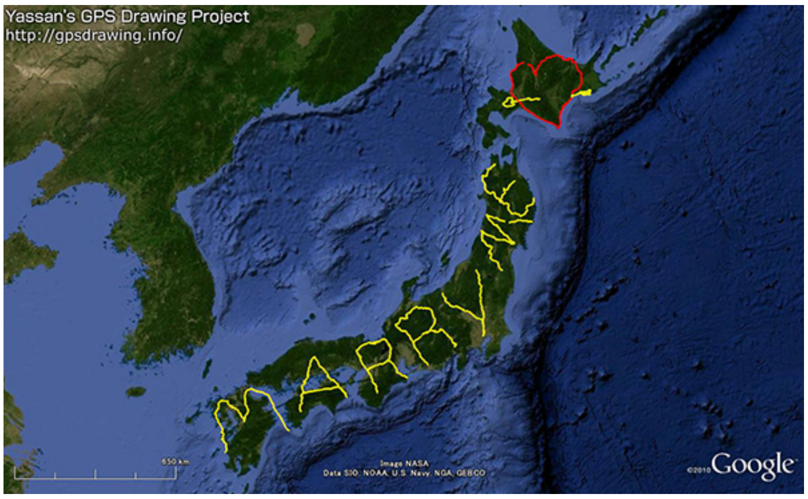


Figure 2: A map of the route walked by a GPS artist to write his marriage proposal

1.1 Project Motivation

Physical inactivity is a modifiable risk factor for cardiovascular disease and a widening variety of other chronic diseases [7]. Data show that one in four adults globally (28% or 1.4 billion people) are physically inactive [8]. The absence of motivation is a factor that contributes to physical inactivity. This project aims to enhance the enjoyment of physical activity by integrating artistic elements into exercise. However, the application is not exclusively for exercise, as there will be an option to drive the route. The World Health Organisation recommends that adults accumulate 150 minutes of moderate-intensity activity per week [9]. By using this application, a user can develop and walk a route that can help them achieve their weekly exercise goal.

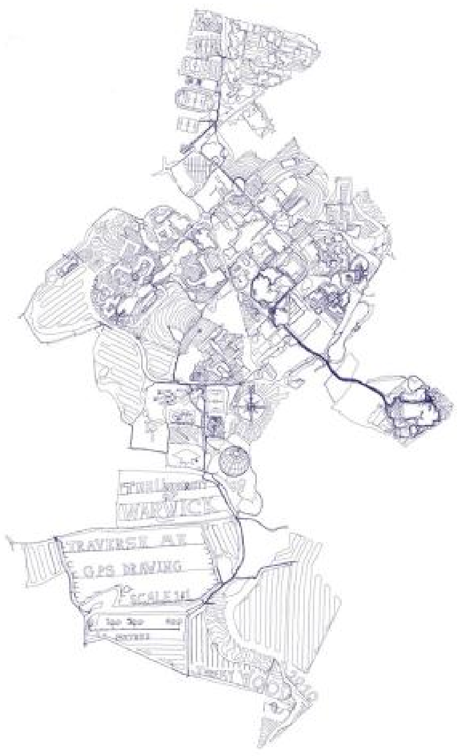


Figure 3: The University of Warwick campus map drawn by Jason Wood

1.2 Objectives

The project aims to create a web application to solve the following problem: Given a range distance about a geographical location, find a route that most accurately corresponds to a user's two-dimensional drawing using only publicly accessible paths. The user will receive an evaluation of the correspondence between the proposed route and the two-dimensional shape. The network topology will change depending on the mode of transport chosen by the user. There will be an option to export the resulting route in a standard format so it can be used in third-party software.

The web application will take two main inputs from the user: the shape they want the GPS art to resemble in the form of a mouse-drawn sketch; The second input will be gathered by placing a marker on an interactive map, and the geographical location will be the rough position of where the route starts. Secondary inputs include the type of transport used, for example, cycling, walking or road network. The other secondary input is the route length. This is because the user will have time and distance constraints. For example, it is not realistic to walk 120km; however, this will be an acceptable distance to cycle.

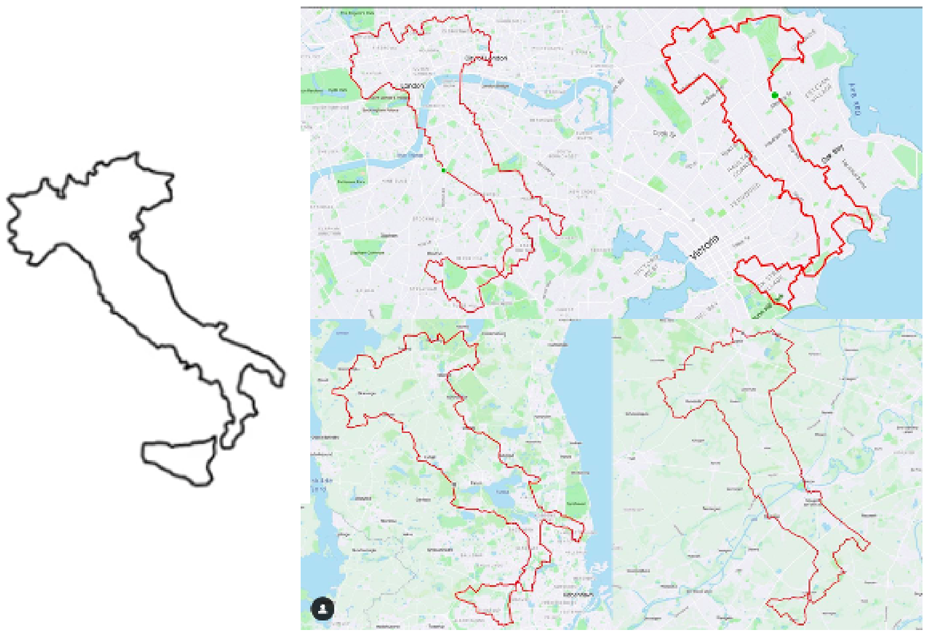


Figure 4: Example routes resembling the sketch of Italy

Figure 4 shows an input sketch, in this case, the outline of mainland Italy, and examples of how routes traversed in different locations can resemble the sketch. Each path is different due to the constraining geographical limitations; however, it is clear that each path resembles the shape of Italy. The project's goal is, precisely as shown in the figure, to generate a route in the chosen location that resembles the user's drawing.

2 Background

2.1 Existing Literature

OSMnx

OSMnx is a Python library that simplifies the process of collecting data, creating, and analysing street networks. The OSMnx package leverages the NetworkX, matplotlib, and geopandas Python libraries to deliver powerful network analysis features, simple visualisations, and efficient spatial querying capabilities [10]. The article discusses all features and provides a comprehensive overview of the OSMnx package and the reasoning behind adding each feature. The OSMnx library has five primary features:

- Acquiring political boundaries and building footprints
- Downloading and constructing street networks
- Correcting and simplifying network topology
- Saving street networks to disk
- Analysing street networks

The feature most relevant to this project is the **analysis of street networks**. The OSMnx package can be used to generate graphs and visualise street networks and route maps. The library can also calculate robust network statistics, such as spatial metrics and street network connectivity. Useful analysis features of this library include fast routing algorithms, calculation of network metrics, and finding the nearest edge to a point.

The **network construction** features are also advantageous as they can extract a road topology within a distance of a given location or the street network of a named place (for example, a town or city). The library also provides functionality to specify different network types to clarify the kind of edges included in the network. For example, the network edges can be specified as those that can be driven, cycled, or walked.

Procrustes Analysis

An integral part of the problem is to identify a measure of correspondence between the route generated and the intended sketch. The route and sketch can be treated as two-dimensional shapes, where a shape is defined as the geometrical data that remains when location, scale, and rotational effects are removed from the object [11]. A shape can be represented as a set of data points where each point corresponds to a vertex of the shape. A Procrustes transformation finds the optimal rotation, scaling, and translation that minimises the distance between two sets of data points [12]. It involves three main steps:

- **Translation:** Both sets of data points are translated so that their centroids coincide, effectively aligning them in space.
- **Scaling:** The scaled data points are then adjusted to have the same scale, typically by scaling them to have the same root mean square distance from their centroids.
- **Rotation:** Finally, the scaled data points are rotated to minimise the difference between the two sets of data points, typically using a method such as singular value decomposition.

The Procrustes distance, d, between two point-sets, P and O, is defined as the sum of squared distances between corresponding points.

$$d(P,O) = \sum_{i=1}^{N_{PO}} \left\| \frac{(p_i - \mu_P)}{\sigma_P} - R(\theta)\frac{(o_i - \mu_O)}{\sigma_O} \right\|^2 \tag{1}$$

Where $R(\theta)$ is the rotation matrix applied to point-set O. μ_P and μ_O are the means of point-sets P and O respectively, σ_P and σ_O are the standard deviations of the distance of a point to the mean of the shapes and N_{PO} is the number of point correspondences between the point-sets P and O.

Sensitivity to Outliers: Squaring the distances can amplify the impact of outliers, as the squared distance between an outlier and the other points can become disproportionately larger. This can result in inaccurate similarity measures, especially in the case of this project, as it is very likely the route will diverge from the perfect path at some point.

Arbitrary Landmark Correspondence: Procrustes distance relies on finding corresponding landmarks between shapes. However, the process of identifying corresponding points is difficult, and different choices of landmarks can lead to different similarity measures [13].

A similar approach to the Procrustes transformation will be used in the algorithm. Using some combination of transformations such as translation, scaling, and rotation to minimise a cost function. The function used to evaluate the Procrustes distance will be different, as squaring the distance is not an appropriate metric in the context of this project. A suitable method to find corresponding points between the sketch and route must be established for this approach to work.

2.2 Existing Solutions

Trace

Trace is a mobile application designed to "explore the possibility of emphasising guided wandering over precise, destination-oriented navigation" [14][15]. To create a 'Trace', the user draws a closed-loop diagram. Next, the user selects the duration they wish to walk, and the application calculates a route and plots it onto the map. The application uses the MapKit SDK to calculate a route with a three-step algorithm:

- Step 1: Determine the starting and ending location (latitude and longitude coordinates) based on a person's GPS position.
- Step 2: Choose a scale that adjusts the size of the drawing over the map based on the intended duration.

- Step 3: Take the projected picture on the map and "snap" each point onto roads.

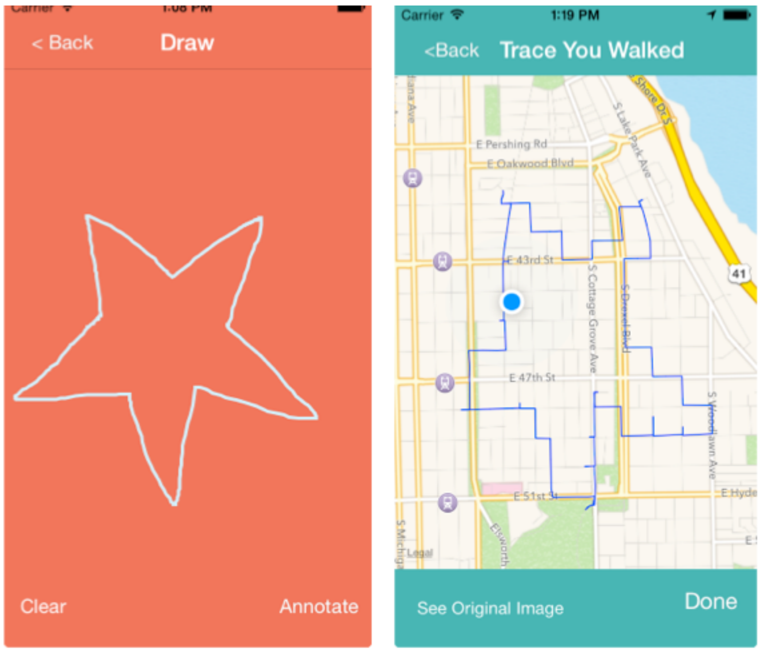


Figure 5: Plotting a star-shaped path in the Trace mobile application

This algorithm usually results in a route with low similarity to the original sketch, and usually, roads are repeated, see figure 5. However, the route still resembles the original sketch. The low similarity comes from the fixed starting point and fixed route distance, so only one route is considered.

The Trace application is no longer publicly available, and no software currently provides the same functionality.

How will the application be similar to Trace?
Both programs accept a user's sketch as input and produce a route to fit the sketch. Both applications show the route projected onto a map, but the Trace app only reveals the path once it has been fully traced.

How will the application be different to Trace?
Trace is a mobile application, whilst this project will be a web application. The Trace application provides the directions to 'trace' the route and only displays the route upon completion. The project will give the user the resulting route and will not provide the functionality to follow the route; the user will have to follow the route and track it using third-party software, for example, Strava. The main difference between the two applications is the algorithm used to generate the route. Trace application uses a naive algorithm that forces the route's starting point at the user's current location. Also, Trace's algorithm does not allow for rotation and only considers a route of fixed length. The project's objective is to create an algorithm that considers various starting points, rotations, and path lengths to achieve the best possible match with the user's sketch. Trace does not evaluate the correspondence between the route and the original sketch, whereas the web application will include evaluation statistics on the generated route.

Strava Route Drawing

Strava is an internet service for tracking physical exercise and incorporates social media-like features, such as befriending other users and liking their activities [16]. It is predominately used by runners and cyclists but has an extensive list of available activities. Strava is the main medium in which GPS art is created and shared. This is due to Strava containing both tracking and social media services.

One feature of the Strava mobile application is the functionality to hand draw routes, similar to that of the Trace application mentioned previously. This option is available to Strava Premium users, which incur a monthly or yearly subscription fee. The feature works by prompting the user to draw the route on top of a map and generates a route based on their input.

3 Project Requirements

Without considering both functional and non-functional requirements, it is difficult to ensure that the system will meet the needs of its users. Functional requirements are essential for defining the basic capabilities of a system, whilst non-functional requirements are concerned with the quality and performance characteristics of the system.

3.1 Functional Requirements

R1 The system **must** extract a road topology within the specified range about a given geographical location from a complete and reliable data set.

R2 The program **must** filter the data set to consider only paths accessible by the user using their desired mode of transport.

R3 There **must** be an interface for the user to sketch a two-dimensional shape and the image stored using an appropriate data structure.

R4 The system **must** calculate a route using only paths from the road topology that closely resembles the two-dimensional shape given as an input.

R5 The interface **should** display an evaluation of the correspondence of the two shapes and give a breakdown of the route, for example, the length and duration of the route for the given mode of transport.

R6 There **must** be an option for the user to export the proposed route in a standardised GIS format ready for the user to download and use on their GPS device.

3.2 Non-Functional Requirements

The project will not be publicly released. The dissertation's primary focus is on designing the algorithm rather than the deployment and delivery of the website. Therefore, the main focus is to deliver the functional requirements;

however, it is still essential to consider the qualities and constraints the system must meet to be considered acceptable.

R1 The system **should** calculate a route within an appropriate time, ideally less than 1 minute.

R2 The code **must** be well-documented and organised to ensure it is easy for other programmers to review.

R3 The system **must** be intuitive and easy for a user to understand.

4 Legal, Social, Ethical, and Political Issues

Throughout the development of the project, it is important to consider potential issues and boundaries the project may pose. As the system involves elements of human interaction, it is crucial to consider the Legal, Social, Ethical and Professional Issues.

4.1 Legal Issues

Should a user decide to follow the route generated by the software, there are a couple of legal issues that may arise. As the route is generated based on information provided by the community-maintained services of OpenStreetMap, it is possible that the data is not up to date or incorrect information has been approved. This could lead to issues with trespassing. Trespassing is the act of entering another person's property without the owner's permission [17]. The algorithm only considers public paths, but if private roads or paths in OpenStreetMap are wrongfully tagged as public, then they will be considered by the program.

There are possible road safety concerns when users intend to traverse their generated route in a car or by bicycle. The route generated may require the user to perform an illegal manoeuvre, such as a U-turn or travel in the wrong direction along a one-way road. This error may be produced from incorrect information from OpenStreetMap or an error in the algorithm. Although the algorithm should only generate legal routes, this has yet to be thoroughly tested. The last issue regarding road safety is the use of a mobile device whilst driving. Rule 149 of the Highway Code states that using a hand-held mobile device or similar device is a criminal offence when driving [18]. Therefore the user should not hold their device at any point whilst driving to follow the route.

The application collects location data from the user. Although this is not necessarily the user's current location, it is likely that the location is nearby. Therefore this raises privacy and data protection concerns. If the application were to be deployed, it must comply with applicable laws and regulations, such as data protection and privacy laws.

4.2 Social Issues

GPS art may have social implications, such as promoting certain cultural or political messages through the artwork generated. It is essential to consider the potential social impact of the art created by the software and make sure it does not promote hate speech, discrimination, or other harmful behaviours. It is difficult to identify and determine the inputs with these intentions as they can be subjective [19].

4.3 Ethical Issues

Suppose the software encourages the users to create GPS art by moving around in a particular area. In that case, it may contribute to increased human activity in those areas, potentially leading to environmental impacts such as increased foot traffic, erosion, or disturbance of wildlife habitats. This is especially the case where artwork would be improved by travelling on public land that is not intended to be trodden. For example, travelling through flower beds would improve the GPS art; however, the flowers would be destroyed in the process.

If a user is not paying adequate attention to their surroundings whilst trying to follow a GPS route to create art, it could pose risks such as accidents, injuries, or other safety hazards to themselves or others. Although the route generated is intended to be followed using third-party navigation software, it is a professional responsibility to ensure that the program does not encourage unsafe behaviour or distract users from their surroundings.

4.4 Professional Issues

The developer is responsible for ensuring the software adheres to professional standards. The majority of the professional issues identified for this project are mentioned in the sections above. In addition, long-term maintenance and support of the project is an issue that needs to be considered if the application is released publicly; however, there is no plan for deployment in the project.

5 Implementation

5.1 Network Extraction

OpenStreetMap
OpenStreetMap (OSM) was launched in 2004 to create an editable world map and was released with an open content license [20]. OSM aims to build and maintain a free editable map database of the world. The main reason to choose OSM is that it is an open-source map and free to use. Although OSM is a community-driven project, it maintains high accuracy and detail and is used by the likes of Amazon, Apple, Facebook, and Strava. While OSM is a valuable resource, it is limited by the need for expanded coverage in specific locations. This is because certain locations have fewer contributors involved in the project. For example, there are less than 0.3 members per million in Madagascar, whereas Italy has more than ten members per million. Therefore, there will

likely be less frequent database updates in areas with fewer users, leading to more inaccuracies.
Data in OpenStreetMap is represented in a simple data structure consisting of three elements: nodes, ways, and relations. A node represents a single point on the map, denoted by a latitude and longitude value. A way is a collection of nodes represented as an open or closed polyline. Ways do not store their own location as the nodes contained within the way store this information. A relation is a data structure that documents the relation between two or more elements (nodes, ways, relations). All data elements can have tags. A tag defines the meaning of each element. A tag consists of a key and a value; for example "highway=motorway" defines the way as a restricted access major divided highway.

OSMnx
The OSMnx Python library can query OpenStreetMaps and retrieve a network within a bounding area and specified tags. For example, figure 6 demonstrates a network of paths publicly accessible by foot at the University of Warwick, extracted using the OSMnx library in Python. All elements within the 1.3km bounding box with a tag stating that the edge is open to the public are included.

5.2 Naive Algorithm

The first step is to build the most basic, fundamental algorithm to generate a route from a sketch. The method employed to calculate this path from a street network is the same approach the Trace mobile application uses. Although the low-level structure may not be the same, the underlying concept is to project the shape onto the street network and 'snap' each point of the sketch onto the road. Further details of the 'snapping' algorithm will be discussed in the following section. It is essential to develop this static generation as it is a point of reference for later iterations of the algorithm.

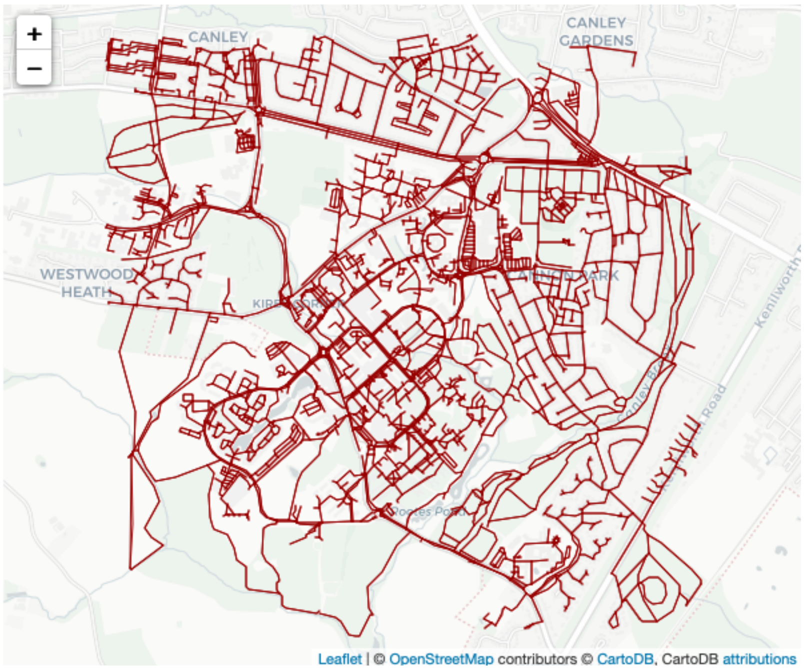

+
−
CANLEY
CANLEY GARDENS
WESTWOOD HEATH
CANNON PARK
Kenilworth Road
Leaflet | © OpenStreetMap contributors © CartoDB, CartoDB attributions

Figure 6: An example walking street network bounded by 1.3km sides centred at the University of Warwick

Sketch Representation

The chosen data structure to represent the sketch was a list of (x, y) coordinates. This allows both closed and open shapes to be represented. This representation method does not allow for continuous curves; however, an accurate approximation can be achieved by frequently interpolating along the curve. Road networks in OSM are represented as a list of nodes containing latitude and longitude. Therefore, approximating curves by interpolating is not an issue, as the final route cannot be a continuous curve.

Throughout the project, the LineString object of the Shapely library [21] is used to represent the sketches. The LineString represents one or more connected line segments by a list of coordinates. The Shapely library does not support smooth curves. However, a LineString can approximate a curve using line segments. Shapely also supports the linear transformations used throughout this project (rotation, translation, scaling).

Sketch Normalisation

The sketch coordinates must be projected into the bounding area of the road network. This is performed by calculating the difference in the maximum and minimum x and y coordinated, essentially the sketch's width and height. The x and y coordinates are then projected into the location bounds, ensuring all points are within the bound using the sketch's centre and width and height. The steps to calculate the projected sketch are as follows:

- **Calculate** the width and height of the bounding area.
- **Normalise** the sketch by subtracting the centroid from all points.
- **Compute** the distance of the coordinates with the x and y coordinates furthest away from the centre.
- **Scale** the sketch in the x and y direction to ensure this coordinate is within the bounds.
- Finally, **translate** the sketch so that its centroid is at the centre of the bounding box.

The resulting coordinates resemble the shape of the original sketch entirely within the bounds of the network area.

Sketch "Snapping"

The basis of this naive algorithm is to snap each line segment of the sketch onto the nearest road on the street network. The first step is to interpolate a certain number of points along the sketch and store these points in a list. The OSMnx library provides a function that returns a list of interpolated points along a LineString at regular intervals. The shorter the distance between interpolated points, the more accurate the algorithm will be, but this increases processing time. The closest node in the street network is calculated for each interpolated coordinate in the projected sketch. There is now a list of the nearest nodes to each point in the sketch. These nodes may not be connected directly by a road.

Therefore for each of the nodes, the shortest path between them is found. By concatenating these shortest routes, a route equivalent to snapping the sketch on the nearest road network is found.

By removing the dead-ends, the accuracy of the algorithm is improved. This is also useful as it is tedious for an athlete to go back on themselves frequently. The route is represented as a list of node IDs. The dead-ends are removed by going through the list and replacing palindromes of length greater than 1 with the first element of the palindrome. Figure **??** demonstrates the steps of the naive algorithm. First, the sketch is projected into the bounds of the street network; then, the shape is snapped onto the road network; finally, the dead ends are removed from the route.

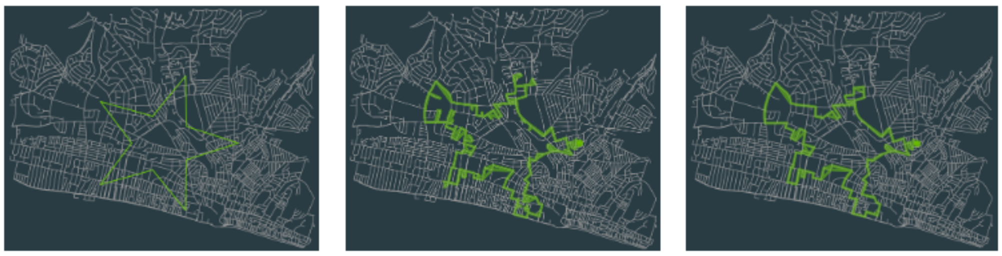


Figure 7: An example of the results from the snapping algorithm and dead-end removal

5.3 Cost Function

A suitable error metric must be established to evaluate the accuracy of a route relative to its original sketch. This error metric is called the cost function, and the goal is to design an algorithm that minimises this value. Many different cost functions can be used, each with its advantages and disadvantages. For example, some cost functions have a demanding computational complexity, and others have trivial examples where they do not work so well.

Corresponding Points

One of the cost functions explored was using corresponding interpolated points along both the sketch and the generated routes. This worked by interpolating the same number of points along the sketch and the route. The first point on the sketch was linked with the first on the route, and so on for the rest of the interpolated points. The average of the distance between these points was calculated, and this is the value used for the cost function.
This method did not work as it was usually the case that the distance between corresponding points was much less towards the beginning and end of the sketch and much more significant towards the middle. This meant that the cost function was not very representative of whether or not the route has good correspondence. Most routes generated have a point where they deviate largely from the actual sketch. If this deviation is near the start or end, interpolated points along the route are much farther apart than on the sketch, making the distance for the rest of the corresponding points large. Consider a route that perfectly matches the sketch but has a significant deviation at the very start of the route. Then the cost function should be minimal as it is an almost perfect route; however, this cost function will be relatively large.

Interpolated Points

Another method was to interpolate n number of points along the generated route and calculate the nearest point on the overlaid sketch and return the average of the distances between these points. At first, it was not apparent in which direction to search for the nearest point (from sketch to route or route to sketch). However, after some tests, it was easy to see that finding the nearest points on the sketch from the route yields much better results. Consider the case where there is a route that perfectly matches the sketch but then diverges massively from the sketch. The closest points from the sketch to the route will all have zero distances, giving a perfect score, although the route does not perfectly match the sketch. Performing the nearest point operation the opposite way achieves a much more useful cost function. By increasing the number of points included along the route, the accuracy of the error is increased. However, this sacrifices the time it takes to compute the cost function, as the nearest point has to be calculated a greater number of times.

Mean Squared Error

There is a decision to make on whether the absolute distance between the points or the squared distance between the points is used to calculate the cost function. By taking the squared error, points further away are penalised much greater than those close. This method is better at finding a route with few large deviations, possibly sacrificing the overall accuracy of the route. Therefore, the squared error was not chosen as a route can usually have large deviations and still look very much like the original sketch.

Error Direction

The direction of the majority of the errors can be calculated by treating each of the distances between the interpolated points as vectors and then adding them together. The sum is calculated by adding the vectors, starting the next vector where the last one ended. After all these vectors have been added, a resulting vector is calculated between the start and endpoints. The direction of this vector is the direction in which most errors occur. The magnitude of this vector represents how many error vectors are in this direction. For example, if half of the error vectors had the same magnitude and pointed south and the other half also had the same magnitude and pointed north, then the resulting magnitude would be zero. Hence, the resulting error vector will have no direction.

Centering The Data

Another explored option was to centre the route before calculating the error using one of the above-mentioned error metrics. Without centring, if the route was a perfect match with the sketch, but the sketch was not directly overlaid on the route, then there would be a positive cost function even though the route is a perfect match. By centring the data in this case, the sketch would be directly overlaid on top of this route. Therefore, using any of the cost functions described above would give a perfect cost. However, this method did not work for the majority of cases and tests. Once again, when a route has a divergence, which is almost inevitable, the centre of the route is skewed. After centring both the sketch and the route, even if the rest of the route is a perfect match, it receives a considerable error value as the divergence has caused the rest of the route to be far away from the respective points on the sketch. Another reason to not centre the data is that it does not provide a direction in which the majority of errors occur relative to their placements before centring. Therefore centring was not chosen to calculate the cost function.

Metric Chosen

After careful consideration, the error metric chosen to evaluate the cost function of the calculation was the use of interpolated points along the route and identifying the distance between these points and the closest points on the sketch. This was due to it giving appropriate values for all cases and the fact that the direction of errors can be utilised to reduce the cost function and ultimately find a better route. So the cost function is a positive number with zero being the best possible value and larger numbers indicating an inaccurate calculation.

An example of how the cost function is calculated can be seen in figure 8. In each image, the star sketch overlaid on the street network is the shape with the red outline, and the calculated route is the green shape. The blue lines represent the shortest distance between the route and the sketch. The blue lines have a start point on one of the interpolated points along the route and an endpoint on the closest part of the red shape. The error metric is then calculated by summing the size of the blue line and dividing it by the number of blue lines. The images in figure 8 show the difference between bad and good calculations. The figure on the left is a bad calculation as there are large parts where the route deviates largely from the sketch, shown where the blue lines are longer. The sketch on the right shows a good calculation, and this can be seen because the route does not deviate from the sketch. Therefore, the left image has a much larger cost function than the right image, proportional to how visually accurate the routes look.

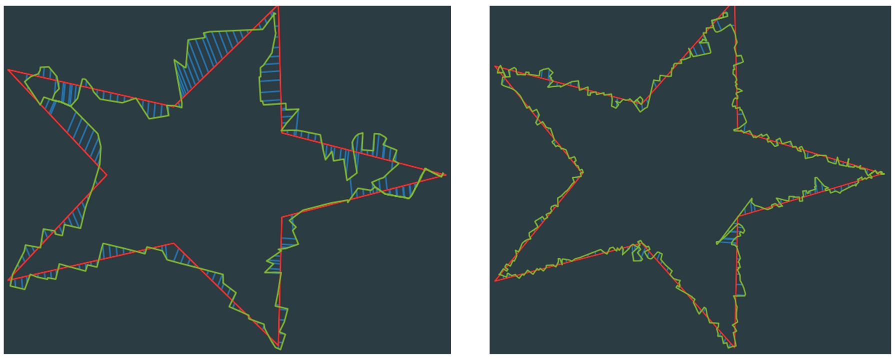


Figure 8: Cost function calculation

5.4 Transformations

When projecting the sketch onto the space bounding the street network, transformations can be utilised to minimise the error achieved when snapping the sketch onto the street network. Only transformations that maintain the original shape of the sketch without warping it can be chosen. The three chosen transformations are rotation, translation and scaling. All are examples of affine transformations [22]. Affine transformations preserve lines and parallelism, ensuring the original sketch is not warped.

- **Rotation** is the circular movement of a shape about a central axis. This transformation will be utilised to match the orientation of the sketch with the street network to increase accuracy.

- **Translation** is the process of moving every point in a shape by the same distance in a given direction. By translating the overlaying sketch in the direction of the majority of errors, this transformation will increase the accuracy of calculated routes.

- **Scaling** is the enlargement of a shape by some scale factor that is the same in all directions. By analysing the error along each line segment of a sketch, scaling about a point with the least error across adjacent lines can minimise the cost function.

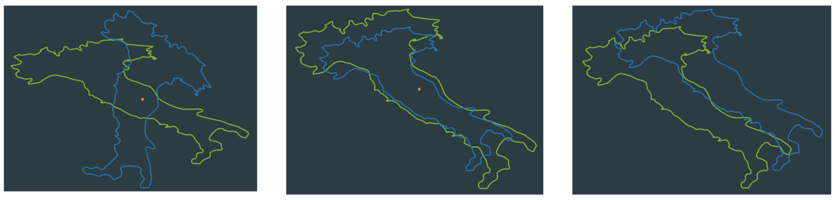


Figure 9: Examples of rotation, scaling, and translation

Rotation

The rotation transformation is utilised by calculating a rotation that minimises the difference between the spatial orientation of the street network and the sketch. The spatial orientation of the street network is calculated as follows. Each road segment in the street network is allocated a bearing depending on the direction of the road segment. Each road network is then placed into a bin depending on its bearing. For example, if there are 36 bins, there would be a bin for all road segments that have a bearing between 355 and 005 and then another for 005 to 015 and so on. These bins can then be combined to form a radial histogram, as seen in figure 10.

A modern city is much more likely to have a street network orientation, with most roads grouped at some 90-degree interval, Whereas a historic town is likely to have a uniform distribution. Each of the distributions shown in figure 10 have a symmetry about the 180 degrees axis. This is because each road segment can be traversed in both directions; hence when adding a road segment to a bin, the road segment is also added to the bin representing 180 degrees further. In the case where a road segment can only be traversed in one direction, for example, a one-way road, then the road segment is only added to one bin and not the 180 degrees symmetrically respective bin. Figure 11 shows example street networks with their respective street orientation radial histogram. Manhattan has a network with almost all roads orthogonal to each other, whereas Boston has a more uniform network orientation.

The orientation distribution of a sketch can also be calculated using the same technique. The goal is to find a rotation that minimises the difference in orientation distribution between the sketch and the street network. This is done by normalising each distribution such that the sum of the weights allocated to each bin is equal to one. Then, for each rotation, the difference in the distribution between each bin is calculated. After calculation, the rotation with the least error is chosen.

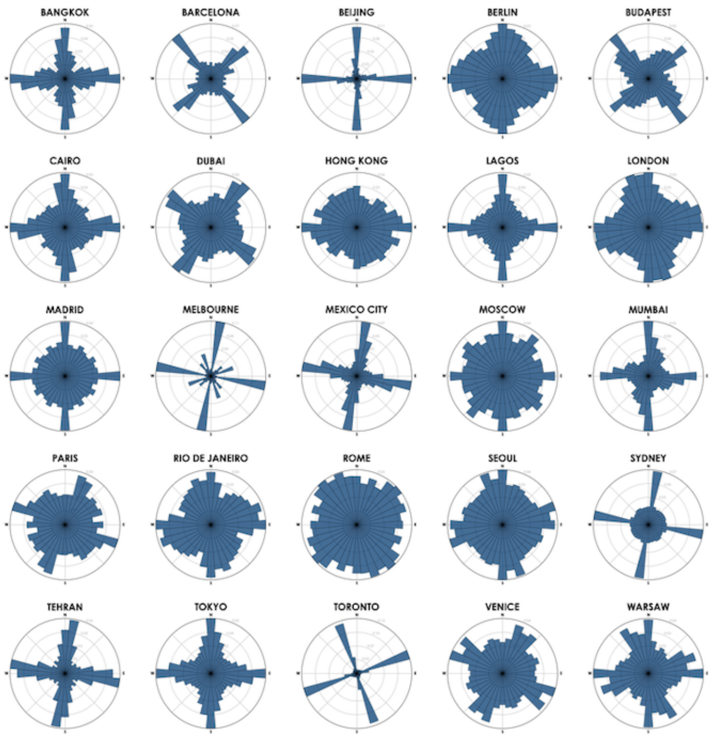


Figure 10: City street network orientations

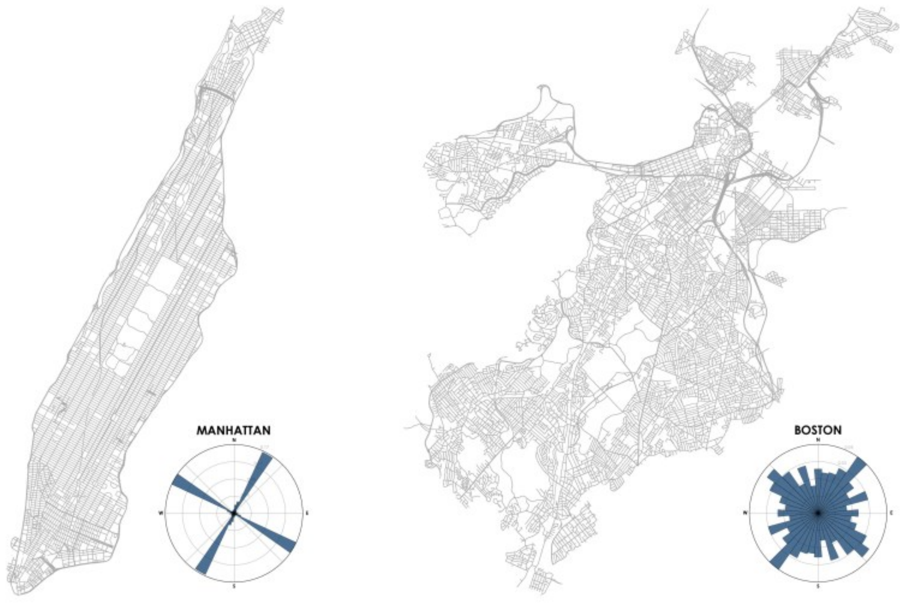


Figure 11: Example street networks and their respective radial histograms

Translation

Translating the sketch in the direction of the majority of errors and recalculating the route using the snapping algorithm with the sketch's new position provides a consistent method of reducing the cost function. When calculating the cost function, the direction of the sum of the error vectors is used to determine the direction the sketch should move to maximise the routeing accuracy. The error direction is represented by two distances, one in the x direction and the other in the y direction. The size of the error vector also represents the number of error vectors in that direction. For example, the size of the vector will be greater if all of the individual error vectors point in the North-West direction. If all of the individual error vectors point in all directions uniformly, then the size of the majority error vector will be minimal. The vector size can then be used to determine how far the sketch should be translated.

This translation is performed iteratively, with the intention of reducing the cost function after each iteration to a point where the accuracy can no longer be improved. The cost function is calculated in each iteration to calculate the direction of the majority of errors and evaluate the accuracy of the route. The best accuracy is maintained over all iterations as well as its corresponding sketch location. After all iterations, the route with the best accuracy is returned, and this is not necessarily the route calculated in the last iteration. Storing the best sketch location over all iterations does not use much space. This is because only the sketch's start location is required, and from this, the rest of the sketch can be calculated and then the route achieved from snapping the sketch at this location. This is a better option than storing the best route, as the route can contain a long list of nodes that takes up too much space. A downside of using this method is the processing time required to calculate the route from the best sketch location. However, this time is short with respect to the whole calculation process.

There was a decision to make when it came to calculating the direction of the majority of errors. When summing the individual error vectors, either each vector has the same weight or has a weight proportional to the size of the vector. Both calculations yield a different direction. The chosen method was to weight each vector in accordance with its size. There were many reasons behind this decision. One was that weighting by size ensures that the translation direction is influenced more by the error direction at large divergences in the route. So this method is more likely to remove significant divergences. Weighting each error vector equally also does not work well when the individual error vectors are small. This is because the translation of points on the sketch that are very close to the route does not matter as they will be moved the same distance away from the route. However, their direction vector is added to the sum even though it is a minimal distance. One way to fix this issue was to add the same weight for all vectors over a certain threshold. However, this method was complex as the threshold needed to be determined, and it would be different for all routes calculated. Therefore, the chosen method was to weight each error vector according to its magnitude.

It is not guaranteed that moving the shape in the direction of most errors will decrease the cost function, even when translating the sketch by an arbitrarily small distance. Consider the case of a single error vector of size ten in the north direction, and the other five vectors are of size one in the south direction. The resulting error vector will be pointing north with size five. Now let us assume that the new route matches the points on the sketch to the same ones on the route. Then the cost function will be greater as there are now five points that are x further away from before and one point that is x closer to where it was before. So the majority of shapes are further away from where they were before giving a larger cost function.

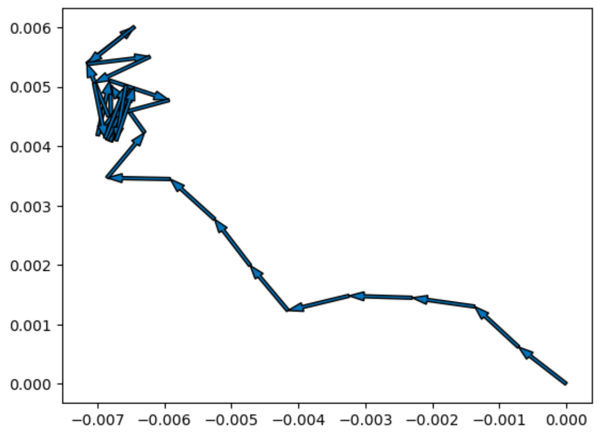


Figure 12: The distance in latitude and longitude the sketch is translated after each iteration

Translation Results

Figure 12 shows the direction in which the sketch moves after 30 iterations of the translation operation. The translation direction is calculated using the sum of all error vectors. The direction after the first iteration is from the (0,0) point on the graph (bottom right). For the first ten iterations, the sketch is translated in the positive latitude direction and the negative longitude direction. After these initial iterations, the error vectors have no general direction and stay at the same point. This is because there is a local minimum at this point. At this point, the sketch's location has an error value that is lower than all of the

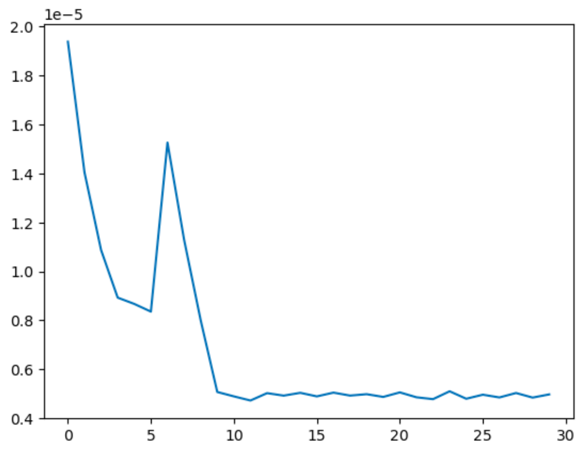


Figure 13: The value of the cost function after each iteration

near surrounding areas, so by moving the sketch a little out of this location, the majority of the error will point back to where it just came from. There is no reason to carry out any more iterations, as the sketch will continue to be translated around this area.

Figure 13 shows the cost function after each iteration. The first entry is the error metric without applying a transformation. The cost function can be seen to decrease after each iteration, with a slight increase on the seventh iteration. After the tenth iteration, where the sketch is in the local minimum, the cost function no longer decreases as the sketches stay roughly in the same place.

Figure 14 shows the positions of the sketches overlaid onto the road network after each iteration. The blue shape is the original position, and where the sketches are placed onto each other is the location of the local minimum. After each iteration, the sketches move in the corresponding direction of the arrow seen in figure 12. After ten iterations, the cost function has been improved, showing that this approach is a valid method of increasing the route accuracy. The issue with this method is that local minima cannot be escaped; incorporating an element of randomness to utilise other transformations like scaling can get the shape out of a local minimum to search for different local minima.

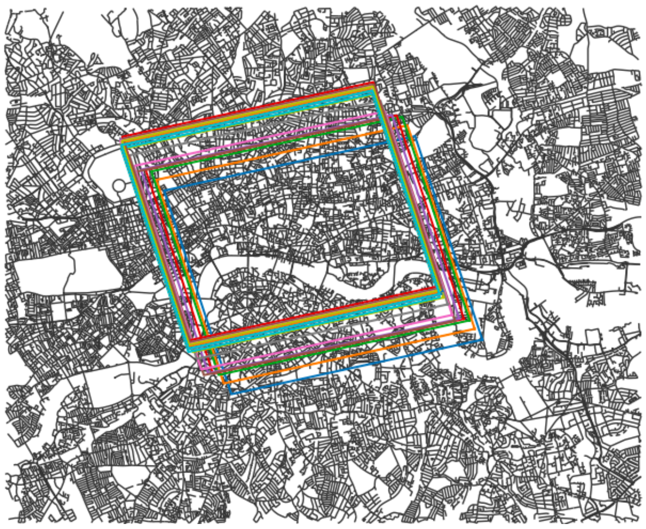


Figure 14: The position of the sketch overlaid onto the street network after each iteration

Scaling

Scaling is the last transformation utilised to increase the accuracy of the route calculated. The intuition behind using the scaling transformation was to keep the line segments with a small error and change the position of the line segments with a large error. This can be done by enlarging the sketch about a corner adjacent to two line segments with low error.

In order to calculate the point at which the sketch is scaled, the error associated with each line segment must be calculated. This is done in the same fashion as calculating the cost function. Each line segment is given an error metric corresponding to the average error along the line. Then each corner is given a value equal to the sum of the error metrics for the two adjacent lines. The corners with the lowest values are the points the scaling will be about. Once one of the corners is selected, a scaling factor ranging from 0.85 to 1.15 is chosen. A new sketch and the new route are calculated for each scaling factor. The chosen route is the one with the smallest cost function; the process is then repeated.

This transformation does not yield results as good as the previously mentioned transformation; however, it still reduces the cost function and finds a slightly better route. This step also does not work well for complex sketches (sketches with more than ten corners). This is because the improvement from the scaling will only be on two of the lines, and when the shape has more than eleven edges, this improvement is small with respect to the shape as a whole. When the shape has many edges, the improvements are not much better than scaling about a random point.

Another problem with using the scaling transformation to improve the accuracy of the route is that it changes the distance and time it takes to complete the route. Therefore, if the user wishes their route to be of a certain length, then this transformation can not be used to increase the accuracy of the route.

Figure 15 provides an example of how scaling can be used to increase the accuracy of the route generated. In the example, the simple sketch of a square is used. The image on the left shows the route generated from the naive algorithm by snapping the sketch onto the nearest roads and deleting dead-ends. The route generated has a large deviation in the bottom left part of the route, caused by the limited available options to cross the river. The goal is to use scaling to remove this deviation. Each line segment is assigned an error metric. The North and West sides of the route have a relatively good accuracy compared to the South and East sides. The corner with the least error between its two adjacent line segments is then chosen as the centre of dilation or enlargement. In this case, the North-West (top left) corner is chosen. Then a scaling factor is applied to the sketch; in this case, the scaling factor is 1.15, and the resulting route can be seen in the image on the right. The blue square shows the original shape, and the orange shape shows the sketch after scaling. The route snapping is the same, where the orange shape overlaps the blue one. This was the goal, as the previously calculated route was already relatively accurate at this point. The new route has also eliminated the divergence in the bottom left corner, increasing the shape's overall accuracy.

Whilst testing how scaling can be used to find an accurate route, a problem arose regarding the cost function. Consider two sketches representing the same shape; however, one is twice as large as the other. Suppose that the larger shape is overlaid onto a street network twice as large as the other one, and both sketches are in the same position. The route generated by snapping each of these sketches will be the same. However, one will be twice as large as the other. As a result, the cost functions for these routes will be different. Although the routes are exactly the same, the scale is different. Therefore, an additional step was required to calculate the cost function. The additional step was to divide the average error between the nearest points by the length of the sketch. This ensures that scaling the sketch does not unfavourably change the cost function. This change will also not affect the cost function used in the translation step.

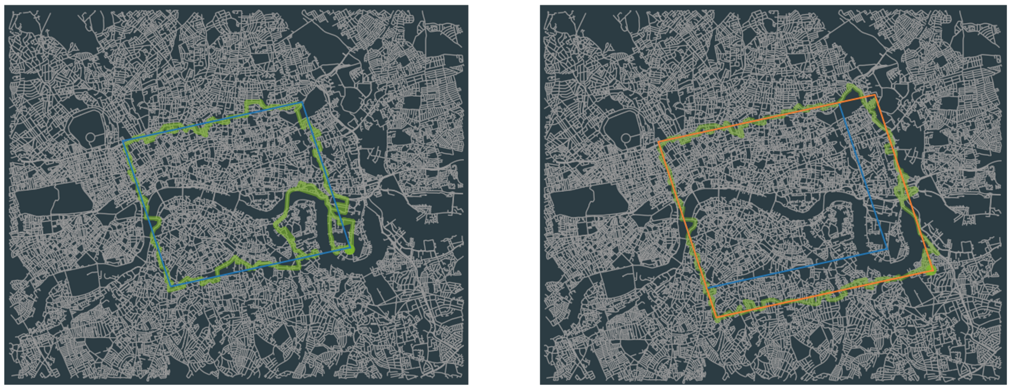


Figure 15: An example of how scaling can improve accuracy

Other Transformations

Reflections and shearings are also examples of affine transformations; however, they were not utilised in the project. A shear transformation is a "linear mapping that displaces each point in a fixed direction by an amount proportional to its signed distance from the line that is parallel to that direction and goes through the origin" [23]. Shearing a figure does not change the area of the figure. The shearing transformation was not utilised as it warps the original sketch and produces a route that is not desirable to the user. Reflection is the only other affine transformation that was not utilised; otherwise, the resulting route differs from how the user wanted it. However, reflection could be used in further development.

5.5 Stochastic Local Search

The final algorithm uses a combination of the three transformations described to calculate a more accurate route than just one of the transformations alone. The basis of the algorithm is to have k sketches placed randomly on the street network that is transformed using the rotation, translation and scaling methods. The route with the lowest cost function is then the route used.
The first step of the algorithm uses the rotation algorithm to align the sketch orientation with the street orientation. K sketches are then generated with this orientation and randomly given a small rotation of 0-5 degrees, either clockwise or anti-clockwise. This is to ensure the K sketches have more variation, the aim being to have more chance of finding a high-accuracy route. Next, each sketch is given a random size and a random location on the street network. From this point, iteratively apply the translation and scaling methods to find a sketch location that gives a route with a low-cost function. When the shape has more than eleven edges, only the translation transformation is used.

5.6 Website Application

The front-end and back-end are two different components that work together to create a complete web application. The front-end refers to the part of the website the user interacts with directly. On the other hand, the back-end refers to the server side of the web application responsible for processing user requests and performing other tasks that are not directly visible to the user.

5.6.1 Front-end

HTML (Hypertext Markup Language) is a markup language used to create and structure the content and layout of web pages. The website only has one web page, and the HTML template defines the different elements on the page.

The elements defined include the buttons, canvas, map, and containers for the different parts of the webpage.

CSS (Cascading Style Sheets) is a stylesheet language used to add styles and formatting to HTML documents, including layout, colours, and other visual elements. CSS was used to place each element in a logical and intuitive position, such as placing buttons relating to the sketch under the canvas. The style sheet was also used to give the website its colour scheme, making it more visually appealing and accessible to people with colour blindness.

JavaScript is a programming language that creates interactive and dynamic functionality in web pages, such as form validation, animations, and handling requests. JavaScript provided the drawing functionality of the website. The canvas element was defined in the HTML template. An event listener was created to get the mouse's location whenever the user clicks inside the canvas. The mouse x and y coordinates are pushed to a list for every click in the canvas. After each click, a line is drawn connecting the last place the mouse clicked and the current place. The first click on the canvas does not draw a line, but it is where the proceeding click will connect. JavaScript also gives functionality to the two buttons relating to the canvas, the "clear" button clears the canvas by deleting all points in the list, and the "submit" button asynchronously sends the sketch information to the back-end.

The interactive map takes the same approach as the canvas by adding an event listener to the map element. The event listener binds a function that clears the existing marker and places a new one in the location of the user's mouse, storing respective latitude and longitude coordinates. A "submit" button sends the location information via an asynchronous request.

5.6.2 Back-end

Flask

Flask, a Python-based web application framework, facilitates the easy development of websites [24]. It allows for web development without worrying about low-level details such as protocols and thread management. As a micro framework, Flask does not include unnecessary components or dependencies, which allows for greater control over the project and minimises overhead. This simplicity was advantageous for the project as it enabled a faster development cycle, allowing more time to focus on the core functionality of generating GPS art. Routing is the process of associating a URL directly to the code responsible for generating the corresponding webpage. Routing is simple in the Flask framework as the **Route()** decorator binds a Python function to a URL or HTTP request. Routing was utilised in this project to seamlessly connect the inputs from the front-end with the Python code responsible for generating GPS art.

The two user inputs received on the website interface are the coordinates of the sketch the user drew and the latitude and the longitude of the marker placed by the user on the interactive map widget. Both of these inputs are posted to the Python code using separate asynchronous requests, which are sent upon activation of a submission button. Asynchronous communication was chosen so the web server could receive data without reloading the page, ensuring the website remained dynamic and interactive. GET and POST are the most commonly used HTTP request methods. In a GET request, the data is contained within the URL, whereas the data in a POST request is sent in the request's body. POST requests were used for both inputs. This decision was forced by the size of GET requests being limited to 2048 characters in most modern browsers [25]. When communicating detailed sketches, the data contained more than 2048 characters.

Once the Python code has all the required inputs, a calculate button sends a HTTP request to start the calculation of the GPS art. Upon calculation, the HTML representation of the Folium map and corresponding route is returned. Upon retrieval of the request's reply, an empty HTML element is replaced with the HTML representation of the route generated as an interactive Leaflet map.

If the user wishes to download the route, they can send a request by clicking the download button, which triggers a Python function to return the file using the in-built flask function **send_file()**.

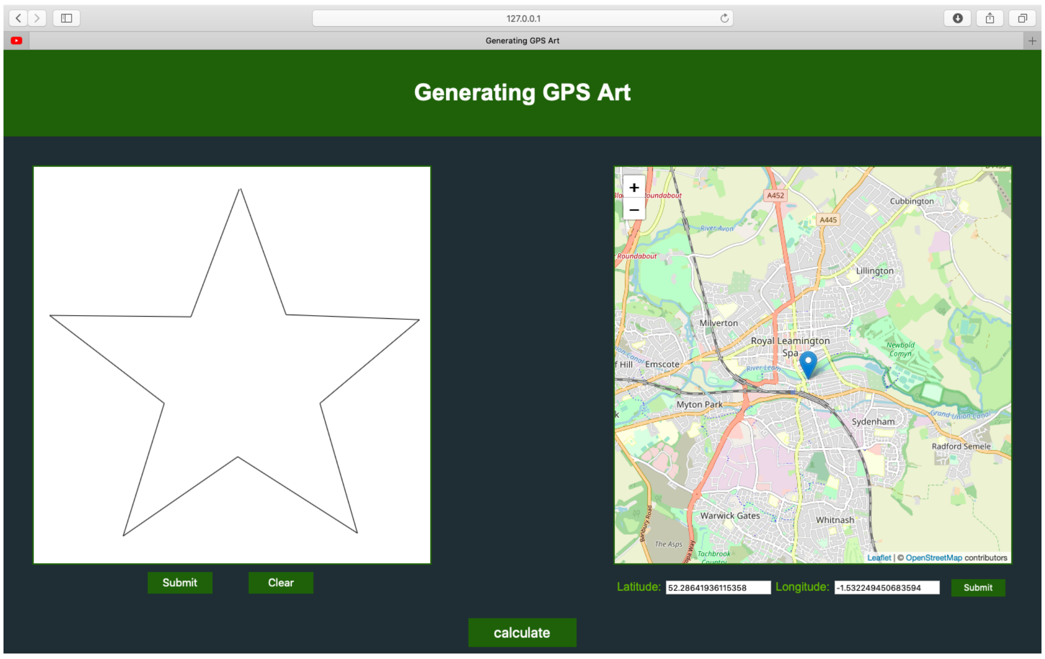
The seamless integration of Flask with Python meant the algorithm developed in Jupyter Notebooks was quickly implemented and ensured smooth communication between the webpage and the algorithm.

Folium
Folium is a Python library that creates Leaflet maps [26]; Leaflet is an open-source JavaScript library used to build web mapping applications [27]. Folium was chosen to display the map as it is easy to generate a base map of specified width and height using a tileset from OpenStreetMap. The OSMnx library also provides functionality to convert a graph representation of a street network into a Folium map. The Folium library has a function which can convert a Folium map into the HTML representation with a leaflet map, which is easy to use with Flask to create the interactive web map.

5.6.3 Website User Interface

The Website interface is minimalistic, ensuring the website is intuitive. The website is separated into two sections vertically. On the left side is a square white canvas where users can input their sketches. The sketch connects the points between where the mouse has been clicked. If the mouse is pressed outside of the canvas, then nothing happens, and the next click will link to the last click inside the canvas. There are two buttons associated with the canvas. The first is a button to clear the canvas in case the user makes a mistake whilst drawing the shape. The second is a button to submit their shape once completed. Both buttons are large and have their function, "Clear" and "Submit", clearly stated in a large font. There is a clear colour contrast between the button and the background, as well as between the button and the button text. The other side of the website contains an interactive map with a marker set at the default location of London. The user can move around, zoom in and out of the map, and drag the marker to their desired location. Once the user is happy with the marker's location, there is a button labelled "Submit" to confirm the location. It is clear that the buttons on the page function as buttons as they change colour when they are hovered over and again when they are clicked on. This keeps the user informed that their actions are being processed.

The website's colour scheme is simple and contains different shades of green, black and white, with the Leaflet map displaying the default OpenStreetMap tileset. The interface was checked against all colourblindness to make sure the application is accessible to everyone.

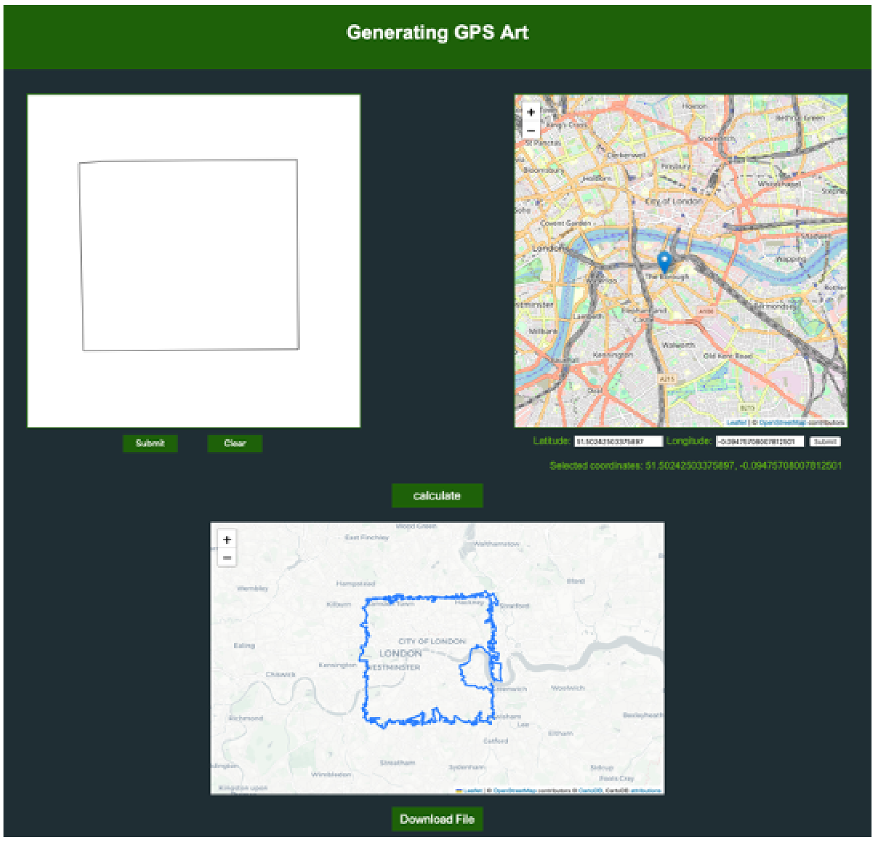


Latitude: 52.28641936115358 Longitude: -1.5322494506835 94

Figure 16: Website user interface

5.7 Route Exportation

The calculated route is represented as a list of nodes on the network that the route passes through; however, third-party software cannot use this format. A GPX file, also known as a GPS Exchange Format file, is an XML-based format to represent routes and can be used by Strava and Garmin-compatible devices. The gpxpy library is a Python library for parsing and manipulating GPX files [28]. Each segment of the generated path must be added to the GPX file as a track to export the route. The track is generated by getting the start and end location of each segment in the route, represented as longitude and latitude coordinates.



Latitude: 51.50242503375897 Longitude: -0.09475708007812501

Selected coordinates: 51.50242503375897, -0.09475708007812501

Figure 17: Website user interface with result

The GPX file can be used on Strava if the user has a premium subscription. "You can create a Strava route from a GPX file by selecting **Dashboard** from the top navigation menu on the Strava website then **My Routes** > **Create New Route**. Next, click the upload button highlighted below and choose the GPX file to create a route." [29] Once the file has been uploaded, the user can edit the route according to their preference. The ability to manually adjust the route provides the option to increase the accuracy of the route.

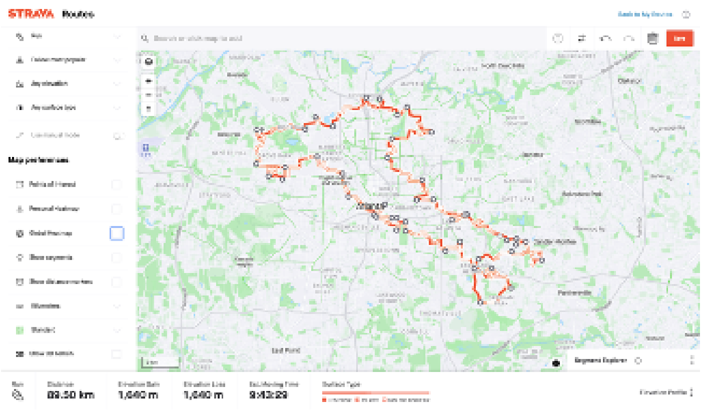


Figure 18: Example of a route exported to the Strava website application

6 Results & Discussion

6.1 Algorithm Accuracy

Simple Shapes

Basic geometric shapes were used to test the accuracy of simple sketches; These included squares, stars, pentagons, hexagons and straight lines. These sketches were tested in multiple locations against the results achieved by the naive algorithm. Different size sketches were used to cover all cases. For all locations and sizes of routes, the algorithm performed much better than the naive algorithm. This was especially the case for routes larger than 15km. The routes generated by the naive algorithm often had large deviations, whereas the project algorithm still had deviations, albeit much smaller deviations. The average deviation distance for the project algorithm remained roughly the same for all sizes of routes. Therefore larger routes had a greater accuracy as the deviation was smaller relative to the rest of the shape. However, this was the same case with the naive algorithm to a much lesser extent. The larger routes generated by the naive algorithm still had significant deviations, even with respect to the size of the route.

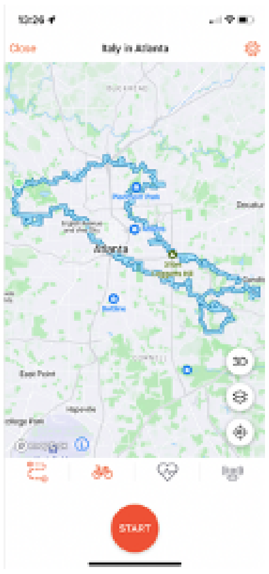
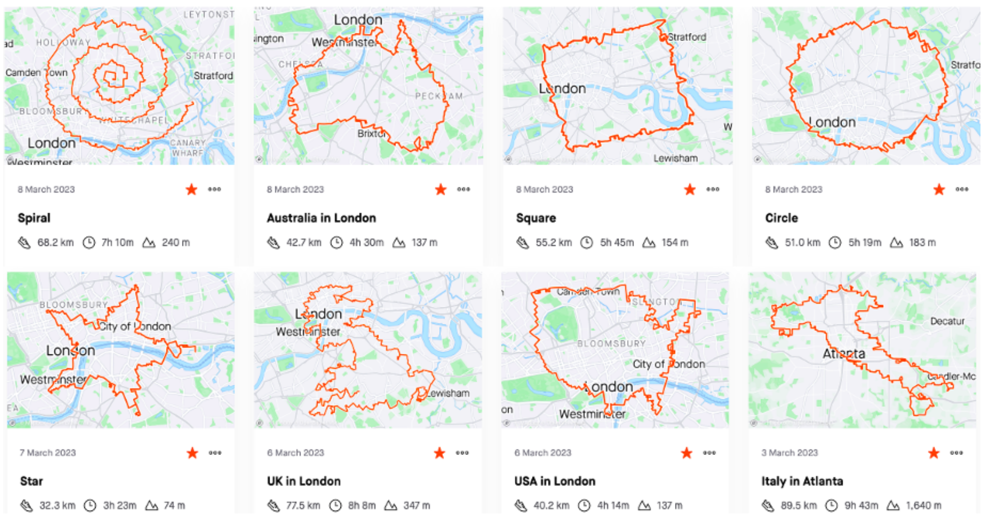


Figure 19: Example of a Strava route as seen on the mobile application

Complex Shapes

The majority of the complicated shapes used to test the algorithm were imported from the Countries database provided by Natural Earth [30]. The shapefile contained the coordinates of the outline of all 258 countries worldwide. By converting the coordinates from the database into a LineString in Python using the pandas library, the shape of the countries can be used in the algorithm. These are complicated shapes as they involve many different vertices and edges of varying lengths. For example, the shape of the United Kingdom without Northern Ireland contained 508 coordinates, and the shape of the United States contained 1990 data points. Spirals and circles were also used in these tests. A Python function was created to output a LineString representation of a circle and another function for a spiral; the parameters for these functions included the number of points and the radius length.
The algorithm's accuracy was tested against the results achieved from the naive algorithm. It was also required to manually evaluate each of the generated routes as, on some occasions, the route would have a cost function much better than the naive route; however, it still did not resemble the original sketch clearly. The routes for the shapes were computed multiple times as the ran-



8 March 2023
Spiral
68.2 km 7h 10m 240 m

8 March 2023
Australia in London
42.7 km 4h 30m 137 m

8 March 2023
Square
55.2 km 5h 45m 154 m

8 March 2023
Circle
51.0 km 5h 19m 183 m

7 March 2023
Star
32.3 km 3h 23m 74 m

6 March 2023
UK in London
77.5 km 8h 8m 347 m

6 March 2023
USA in London
40.2 km 4h 14m 137 m

3 March 2023
Italy in Atlanta
89.5 km 9h 43m 1,640 m

Figure 20: Examples of generated routes downloaded and imported into Strava

domness of the algorithm produces different routes with the same inputs. The routes were also computed at various locations with a range of network layouts, for example, urban and rural networks. The naive algorithm struggled to represent the complex shapes accurately; this is due to the complex shapes having more detail. The routes generated by the project algorithm had a clear correspondence to the original sketch when the distance of the route was a medium to long length. However, smaller routes could have been approximated better. This is because the distance of the edges on the sketch that is overlaid onto the street networks is smaller than the road segments on the network. So it is tricky or impossible to have a strong correspondence. Generally, the algorithm performed very well, but there are some limitations that will be explored in a later chapter.

Figure 21 shows the difference in accuracy between the routes generated by the final algorithm and the naive algorithm imported into Strava. The results show a clear difference in accuracy, the naive algorithm routes tend to have much more significant deviations, especially where there is a geographical restriction, such as a river.

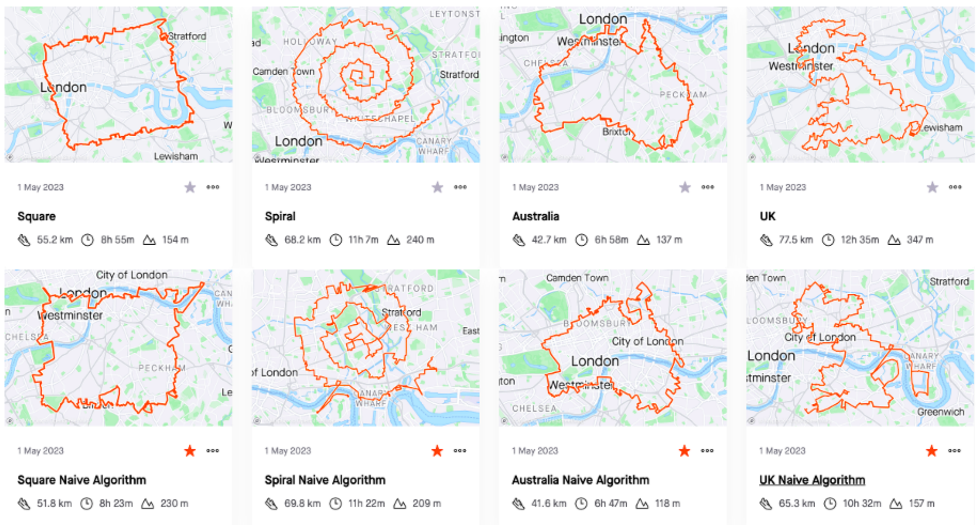


Figure 21: A comparison of routes generated by the final algorithm (top row) against routes calculated by the naive algorithm (bottom row)

6.2 User Trial

A trial run of how the software is intended to be utilised was performed to see if the routes generated by the algorithm are viable. The trial's main purpose was to see if the exported route could be traversed in its entirety and to see if the resulting path traced resembled the user's original sketch. The sketch chosen in the trial run was a star, and the location was Royal Leamington Spa, United Kingdom. A star is a relatively simple shape used previously in GPS arts. Royal Leamington Spa has a relatively dense street network; however, there are some geographical constraints in the form of a river, train line, and canal, all with limited crossing points. Royal Leamington Spa is reflective of almost all medium to large cities worldwide. The mode of transport was chosen to be walking as it was the easiest method to conduct the trial with.

The route generated by the algorithm can be seen in Figure 22. For the most part, the route resembled the original sketch of the star; however, the West part of the shape diverged from the original sketch a lot more than the rest of the sketch. The route was exported as a GPX file and imported into the Strava web application. Upon a brief analysis of the route, it was clear that the divergence on the west side of the sketch could be changed with a divergence that would go further to the East, slightly increasing the cost function of the shape, however, making it more pleasing to the human eye. The before and after versions can be seen in figure 23. The rest of the route remained the same. The route had a distance of 11.51km, an estimated walking time of 2 hours and 6 minutes, and an estimated running time of 1 hour and 12 minutes. Once the route had been saved, it was sent to the GPS watch and was ready to be followed. The route can be seen on Garmin[31] and Strava[32].

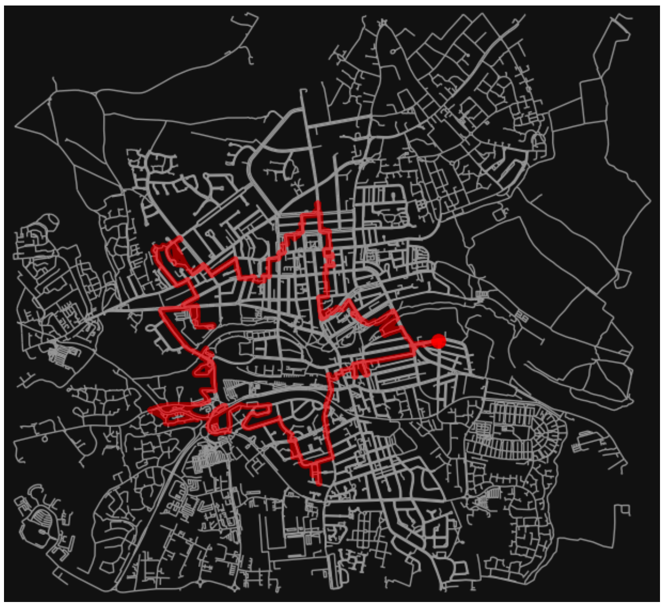
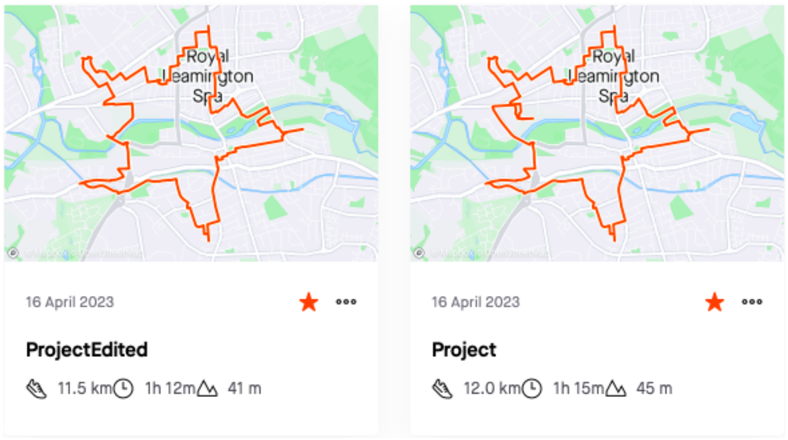


Figure 22: Example route generated by the algorithm

The trial used two GPS devices: a Garmin Forerunner 735XT watch and an iPhone XR using the Strava mobile application. Once at the starting position, both GPS devices were turned on, and the route was followed. For the most part, it was intuitive which roads were intended to be followed; however, there were certain points on the route where it was not clear where the route would go next. This was evident in two places along the route. Two kilometres into the activity, the route goes into a shopping centre. The shopping centre is open



16 April 2023
ProjectEdited
11.5 km 1h 12m 41 m

16 April 2023
Project
12.0 km 1h 15m 45 m

Figure 23: The route after a slight adjustment

to the public, so it did not provide an issue. However, the shopping centre is only open during daytime hours; therefore, the route the user would have to follow could differ depending on the time of day they wish to perform the activity. The second point during the activity that could have caused difficulties was when the route directed the user into an alleyway. The entrance to the alleyway had a half-open gate and was very overgrown throughout its entirety. Due to the circumstances, it needed to be clarified whether the passageway was open to the public. This could have posed a problem if the gate was shut or the user decided it was too overgrown to continue. Should this be the case, a detour would have to be followed, sacrificing the artwork's accuracy.

Once the route had been completed, both GPS devices were turned off, and their routes were saved and uploaded to Garmin[33] and Strava[34] as seen in Figure 25. The activity covered a distance of 10.45km and took 1 hour and 46 minutes.

By hand-drawing the underlying sketch that the algorithm used to generate the route on the Strava mobile application, the resultant route can be compared against the route that was generated by the project. The hand-drawn sketch on the Strava mobile application was a near-perfect representation of the one used by the algorithm. The sketch generated by the mobile application had few resemblances to the GPS art that was generated by the website application; it

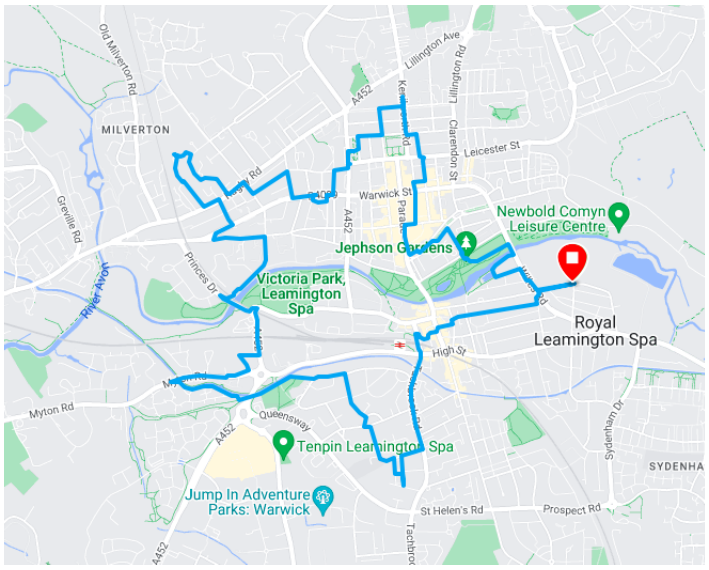

MILVERTON
Victoria Park, Leamington Spa
Jephson Gardens
Newbold Comyn Leisure Centre
Royal Leamington Spa
Tenpin Leamington Spa
Jump In Adventure Parks: Warwick
SYDENHA
River Avon

Figure 24: Final route that is to be followed

is also difficult to identify the route as an imitation of a star.

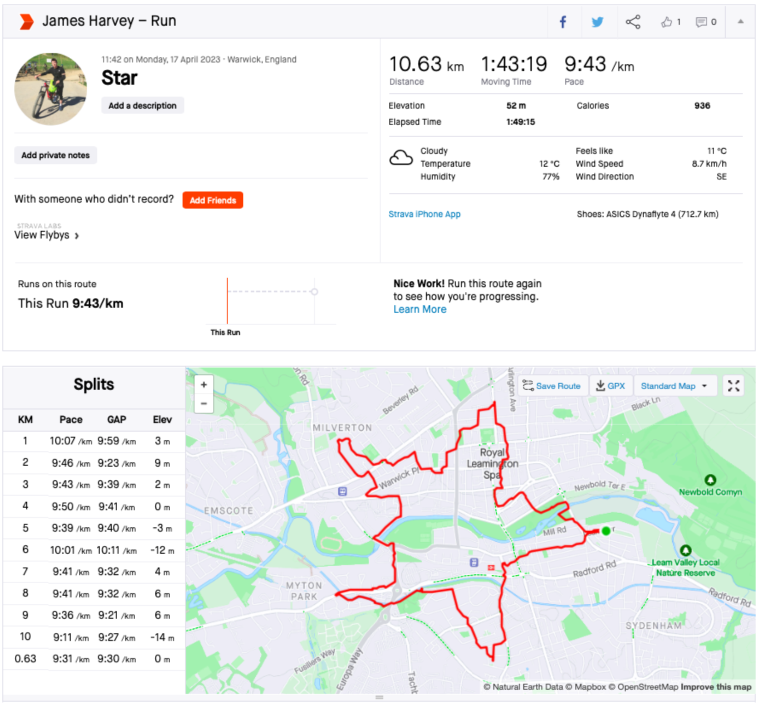
6.3 Requirement Evaluation

6.3.1 Functional Requirements

Regarding the functional requirements mentioned in chapter 3.1, the project was a success. All requirements except requirement **R5**, this includes all of the requirements with the must priority. Requirement **R5** states that the system should provide a breakdown of the route distance, elevation, and duration of the route. This requirement was not implemented as third-party software already provides an in-depth route analysis, which can include a personal estimated moving time [35]. Therefore, it was decided to focus on requirements categorised with the **must** priority.

6.3.2 Non-Functional Requirements

In terms of the non-functional requirements, **R1** and **R2** have been met. However, it is difficult to evaluate the success of **R3** as no user acceptance test has



James Harvey – Run

11:42 on Monday, 17 April 2023 · Warwick, England

Star

Add a description

Add private notes

With someone who didn't record? Add Friends

STRAVA LABS
View Flybys ›

10.63 km	1:43:19	9:43 /km	
Distance	Moving Time	Pace	
Elevation	52 m	Calories	936
Elapsed Time	1:49:15		

Cloudy		Feels like	11 °C
Temperature	12 °C	Wind Speed	8.7 km/h
Humidity	77%	Wind Direction	SE

Strava iPhone App

Shoes: ASICS Dynaflyte 4 (712.7 km)

Runs on this route

This Run **9:43/km**

Nice Work! Run this route again to see how you're progressing.
Learn More

Splits

KM	Pace	GAP	Elev
1	10:07 /km	9:59 /km	3 m
2	9:46 /km	9:23 /km	9 m
3	9:43 /km	9:39 /km	2 m
4	9:50 /km	9:41 /km	0 m
5	9:39 /km	9:40 /km	-3 m
6	10:01 /km	10:11 /km	-12 m
7	9:41 /km	9:32 /km	4 m
8	9:41 /km	9:32 /km	6 m
9	9:36 /km	9:21 /km	6 m
10	9:11 /km	9:27 /km	-14 m
0.63	9:31 /km	9:30 /km	0 m

Figure 25: A walk tracked using Strava on an iPhone XR

been conducted. As the system is minimalistic, and all parts of the interface are clearly labelled, most users will likely be able to navigate the system easily. This requirement can only be fully evaluated after user testing.

6.4 Limitations

Like any research endeavour, this study has limitations. As an algorithm designed to address a complex problem, it is essential to recognise the potential constraints and challenges that may affect the validity and accuracy of the results. This section will provide a transparent discussion of the limitations of the project, highlighting opportunities for further research and improvement.

Network Extraction Time

The street networks are currently downloaded from the internet using the OSMnx library. This takes a large chunk of time on standard computers for

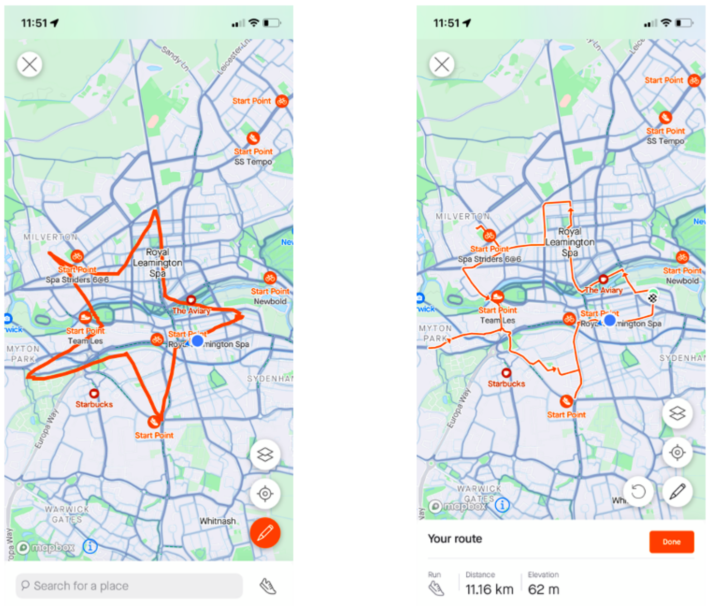


Figure 26: A route generated by the Strava drawing feature

large or dense networks. The time it takes to download the street network is proportional to the number of nodes in the bounding area; therefore, extracting the same area in London will take much longer than taking the same area in the countryside. This is an issue as it means that the software can take an unreasonable time before the algorithm to calculate the best route even starts. However, once the street network is downloaded from the internet, the algorithm can start straight away, so if the user wishes to find a corresponding route for multiple sketches, then this issue is less of a problem.

One method to address this issue is to have the entire planet on the server side, which means there is no time required to download the data from the internet, decreasing the time it takes for a route to be calculated. The problem with choosing this method is that storing these networks is difficult as the file sizes are very large. For example, the latest planet XML file contains 126GB of data [36]. This could be mitigated by only including the necessary data points. This would only include the networks that may be queried, such as roads accessible by pedestrians, bikes or cars. OSMnx has methods that can extract a street network from the internet as well as a file, so this change can be made in future iterations of the software without affecting the rest of the program, so it will be easy to implement.

Unconnected Graph

The chosen area may have no network or limited network for the chosen mode of transport. This can cause an issue if there is no shortest route between two nodes chosen as the closest points during the snapping process. This will cause an error as getting from one side of the network to another is impossible. Therefore, when the network is limited, the algorithm will not always be able to display a valid path. However, plotting a complex sketch manually is likely problematic in this scenario.

Geographical Limitations

Geographical and geopolitical features such as rivers, train lines, and country borders have limited crossing relative to other points on the street network. Features like this prove difficult when calculating an accurate route, especially when these features are at the centre of where the user wishes the route to be. Because there is only a limited place where the route can cross these features, large divergences often occur, reducing the overall accuracy of the route. In future iterations of the algorithms, these limited crossing points could be prioritised as it is likely that divergences will occur because of these points.

An example can be seen in figure 27, where the sketch input is the outline of Italy, and the sketch is to be projected onto the street network of London. The river Thames is in the centre of London, and when calculating the route, the bottom half of the route becomes a lot more skewed than the top part as there are a limited number of places where the route can cross the river. At the point in London where the route approaches the river, there is only one bridge across the river.

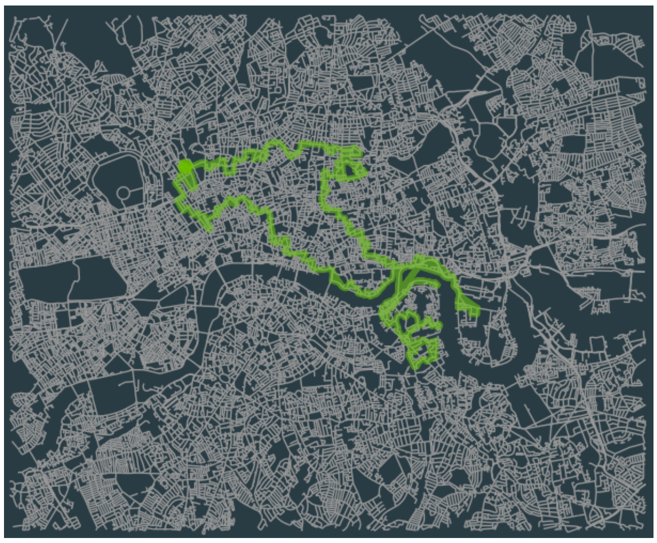


Figure 27: Example of a geographic feature reducing accuracy

Urban vs Rural networks

As rural networks are less dense, there are fewer route options to explore, making it more unlikely that an accurate route can be found. This is especially the case when the user tries to project complicated shapes with detailed features, as a rural network does not have enough route options to capture the details. Routes generated in an urban area are typically much more accurate than their rural counterpart. This is due to there being many more options the path can take. However, the downside of calculating a route in an urban area is that the network's extraction time can take much longer than in rural locations for the same bounding area. Calculating a route in an urban area usually takes longer because there are more nodes. So if there is a computational time limit, the urban area will process fewer possibilities than the rural network.

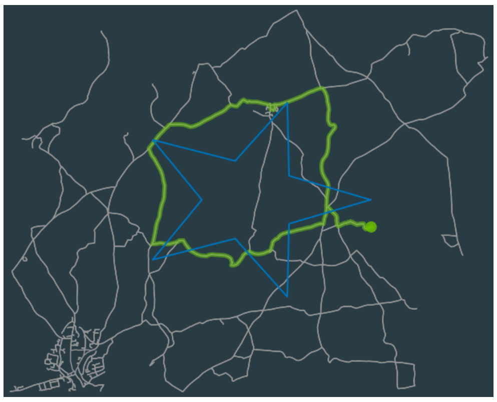


Figure 28: Example route calculated in a rural area

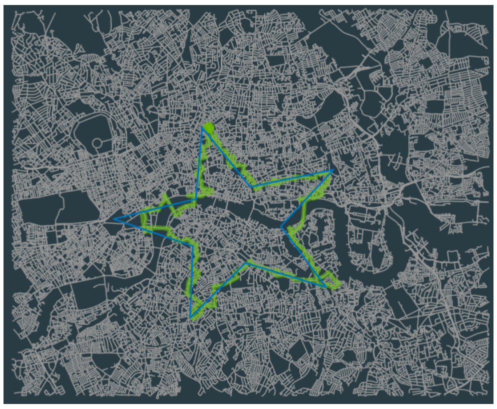


Figure 29: Example route calculated in an urban area

Short routes

Short routes are more likely to be less accurate than large routes. This is because divergences are large relative to the rest of the route, making the resultant path look less like the desired sketch. This is especially the case for complex sketches, as there are not enough path options on the street network to capture the details of the sketch. Simple geometric shapes are more likely to work well if the route is short; however, the accuracy is typically much less than if the route was longer.

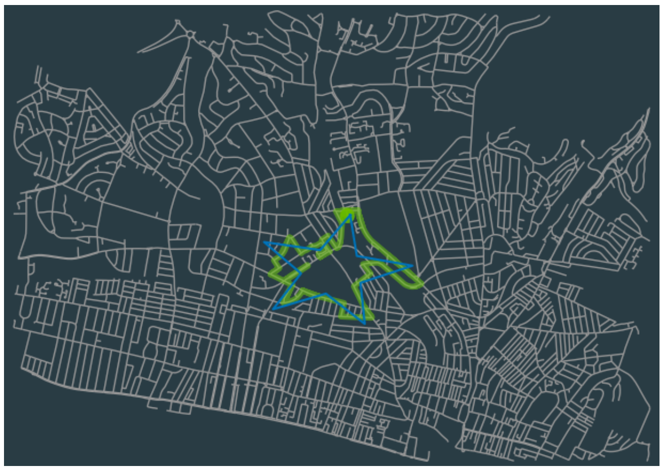


Figure 30: Example of a small route

Snapping Calculation

Sketch snapping is an integral part of the algorithm. The snapping works by interpolating along the sketch and finding a list of all of the nearest points to each of the respective points and then finding the shortest route between these points. There are currently no functions to find the closest points on a road segment in the OSMnx library; instead, the closest node is found. Even if the closest point on a road segment could be found, the OSMnx library does not provide accurate routing options. Using the closest point on the street network instead of the closest node may increase the accuracy of the resulting route.

A library exists to address this problem, called TaxiCab [37]. However, upon personal examination, there were many issues with the implementation, and the library did not provide a reliable resource that could be used in the project. For example, figure 31 shows how finding the shortest route between two points differs when using TaxiCab and OSMnx. The TaxiCab library did not produce reliable results, so this routing approach was not utilised in the project. However, it would be worth exploring further as it may increase the accuracy of the routes generated.

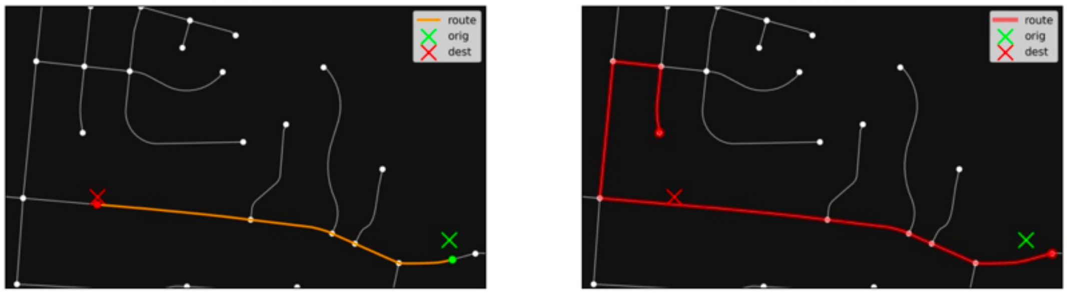


Figure 31: TaxiCab (left) vs OSMnx (right) routing results

One-Way Traffic

If a route contains a one-way road, the direction of traversal will affect the distance of the route. For example, figure 32 shows how the distance of the shortest route between two close points can dramatically change depending on which node you start from. Therefore the snapping part of the algorithm would not work well if the nearest nodes are as seen in the figures.

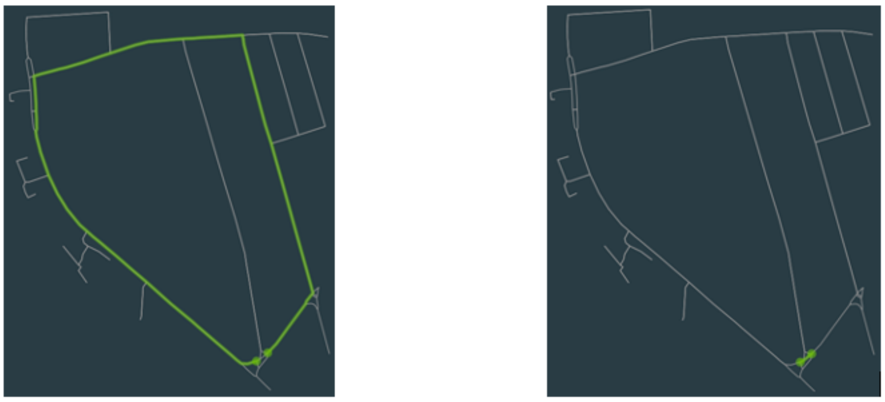


Figure 32: An example of how the direction of travel affects the distance of the shortest route

Dead-end removal

Although the majority of the time, removing dead-ends improves the accuracy of the route, this is not always the case. For example, consider a map with a square with a small piece missing; this will be a dead-end and hence removed, but if it is kept, the accuracy will be much higher. As it is a relatively small number of cases with these issues resolving the issue was not implemented in the first iteration of the software. However, this issue could be fixed by allowing the user to choose whether they wish to include dead-ends. A more complicated algorithm could also be employed that gives a weight of importance to each dead-end and how much its removal affects the cost function. This will come at a higher computational cost, and it may not result in a higher level of accuracy, but it would be worth exploring if the algorithm is to be improved further.

Map Projections

Throughout the project, the geographic coordinate system is used. It is a spherical coordinate system for measuring positions on Earth as longitude and latitude positions. Latitude measures the North-South position on the surface ranging from -90 to 90. Longitude measures the East-West position on the surface ranging from -180 to 180. Due to these definitions, the difference in longitude represents a different distance depending on the current latitude. Therefore projecting a shape from the Cartesian plane to the geographic coordinate system can skew and distort the original shape [38]. This is especially true near the North or South Pole, where all longitude lines converge to the same positions.

Figure 33 shows how coordinates representing a square in the Cartesian plane do not represent a square in the geographic coordinate system. This is because the shape does not have edges of the same length, and the north and south parallel edges are not the same sizes.

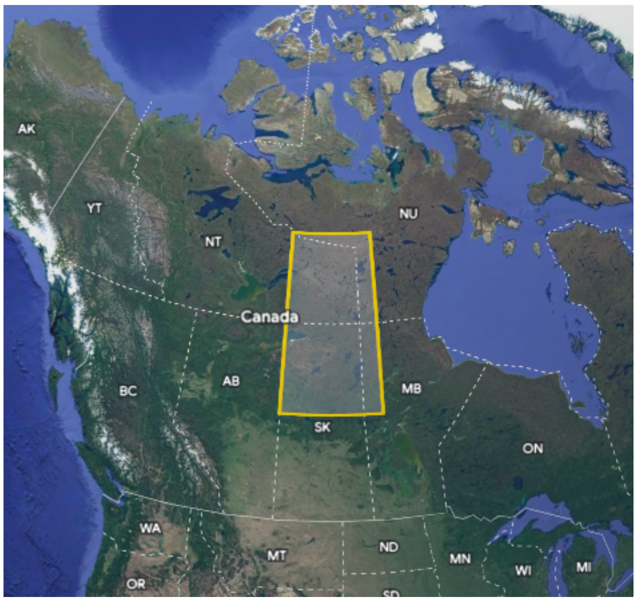


Figure 33: Projection of polygon (-110, 55), (-100, 55), (-100, 65), (-110, 65) onto a map

GPS Device Accuracy

Although it is not a limitation of the software, the accuracy of the GPS device used during the activity can result in a less accurate route than the proposed path. If the accuracy of the GPS device is poor, then the resulting route may have erratic lines as the uncertainty of the GPS is large. The capture rate of a GPS device is the number of times it records its location per unit of time. If the capture rate is low, some details of the actual route may differ from the traversed path.

As GPS devices require signals from satellites, some locations may have poor connections. Examples of where devices cannot receive signals include: indoors, in a tunnel, near tall buildings, underwater, and under trees [39]. When there is poor or no signal, the accuracy of the recorded route may be affected, ultimately affecting the results of the intended GPS art. For example, figure 34 show a GPS device receiving a signal that has reflected off a building. As the GPS device cannot tell if the signal has been reflected, the position of the signal, if it had not been reflected, is recorded. Modern cities have been described as "urban canyons" where getting an accurate GPS reading is difficult [40].

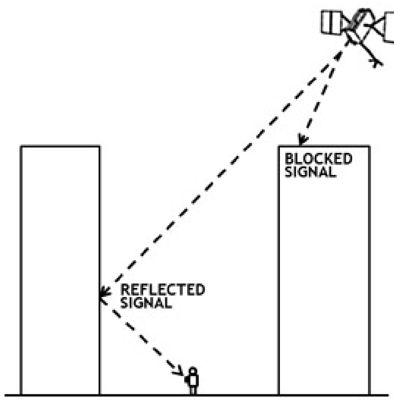


Figure 34: An example of how an inaccurate location can be obtained from a GPS device

7 Project Management

Effective project management is a critical factor in ensuring a successful result. Project management encompasses planning, organising, and controlling resources and tasks to achieve specific goals and deliverables within defined timeframes. Although this is an individual project, it is vital to outline the intended structure to ensure that all deliverables are produced on time and that progress and success are measurable. The following sections will detail the project management tools and methodologies used to ensure the smooth running and success of the project.

7.1 Software Development Methodology

Although the project requirements were clear it remained unclear how they were going to be, until further research was conducted. Therefore, the project lended itself to an agile development methodology. After researching useful tools and possible approaches, sprint cycles were used to iteratively improve the algorithm. A sprint breaks the project into sizable chunks. The majority of the sprints had the goal of improving the accuracy of the algorithm by combining a new approach with the previously developed methods.

7.2 Schedule

The project consisted of three main stages. The first stage was the research, which involved looking into existing solutions and tools and determining the best combination of approaches. The following two were designing the algorithm and developing the website; both stages started after the research was conducted. The algorithm design was performed in parallel with the development of the website, as they mainly consisted of independent parts. Furthermore, each task regarding the algorithm's evolution was treated as a sprint because there were clear goals, and each iteration intended to improve on the last, so a sprint was well suited.

The project schedule can be seen in figure 35. There were several dependencies between activities. One of the dependencies is that the algorithm could not start development before any research on the approaches or tools was conducted. The other dependency comes from the results of the naive algorithm. All iterations of the algorithm involve snapping the sketch to the network after some manipulation; therefore, this task was crucial. The website development also depended on this activity as there was no reason to have a website interface without the basic algorithm.

The work was carried out in long uninterrupted bursts two to three times a week. Efficiency was maximised by working on the project for longer than four hours at a time. This was because setting up the work environment for the algorithm took a long time, and the understanding of the code was fully realised within the first few hours, so programming was more efficient. Each work session was goal-oriented, with specific goals and outcomes to be achieved.

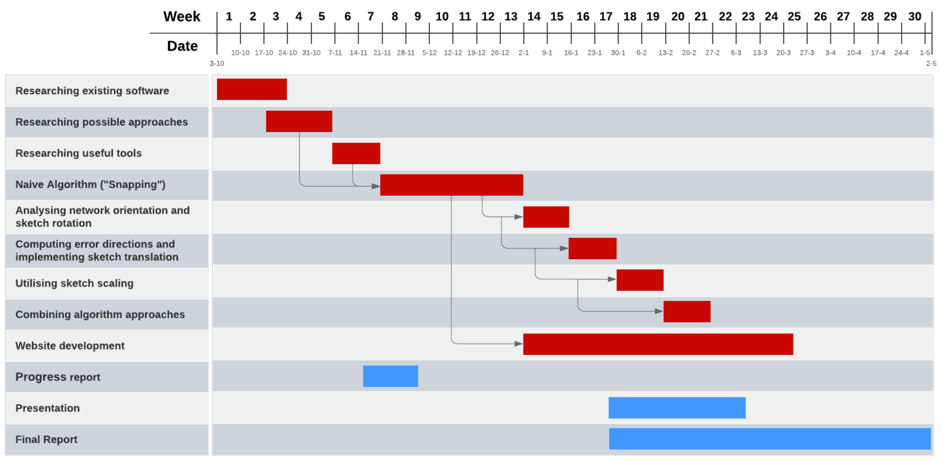


Figure 35: Third year project updated timeline

7.3 Version Control

Git and GitHub:
Git was employed as a distributed version control system to effectively manage changes to the code, data, and documentation. A comprehensive history was maintained to easily revert to previous versions, and compare changes. GitHub allowed the project to be backed up onto the cloud to mitigate against hardware malfunction.

Physical Backups:
There were long periods of limited internet access throughout the project, meaning backups to the cloud were not possible. Significant project progress occurred within these times, therefore a physical copy of the project was maintained during these times.

7.4 Conda Environment

During the development of the algorithm, a Conda python environment [41] was used. A Python virtual environment consists of a certain Python version and some packages. There is a multitude of reasons to use a package manager. One of the reasons was to ensure that packages required by other projects did not interfere. Another reason was to ensure the application works smoothly on all computers if the environment is set up the same. Conda was chosen as the package and environment manager as it is easy to understand its directory structure, and it is very flexible as it contains many packages. A list of third-party dependencies used in the Python environment is maintained in a requirement.txt file.

7.5 Jupyter Notebook

Jupyter Notebook maximised efficiency throughout the project's development and implementation. Jupyter Notebook is a free, open-source, web-based development environment for Python to create documents that contain visualisations. Matplotlib is a widely-used Python library used for data visualisation and has extensive support in Jupyter Notebook. It is also the format in which all plotting functions of the OSMnx library return. Therefore, Jupyter Notebook was a useful tool as street networks and routes calculated can be quickly visualised and compared against other computations and locations.

Jupyter allowed for quick testing, as once the street network was downloaded, the network could be queried without having to re-download. Small parts of the code could be changed without running all the code in the notebook. This was especially helpful when it came to changing the location and shape.

The Jupyter setup lends itself to tidy code. This is because a new cell can be created for each individual part of the code. This aspect is useful for demonstrations and understanding the function of each part of the code.

8 Conclusions

In conclusion, despite the limitations, the project succeeded and provided an original approach to solving a problem with no current existing solutions. The project faced significant technical difficulties throughout its duration; in spite of these challenges, the issues were addressed and resolved where possible to create the final system.

The first part of the project involved researching existing solutions, tools, and methods for similar problems to develop a novel method for generating GPS drawings. Ideas from Procrustes analysis, the Trace mobile application, and the OSMnx library were combined to form a powerful basis for the algorithm to build on. The second half of the project involved iteratively improving the algorithm, using sprint cycles, by implementing different affine transformations to minimise the cost function. In addition, a dynamic website was built in parallel with the algorithm, providing an intuitive interface to interact with the algorithm seamlessly. Finally, robust projected management techniques have been employed to ensure progress remains on schedule and mitigate unforeseen complications.

Although the system meets all the crucial requirements, user testing must be considered in future development to identify limitations, assess usability, and further evaluate algorithm accuracy. Nevertheless, the project has achieved all the objectives and serves as a solid foundation for continued evolution.

8.1 Further Work

Throughout project development it is important to acknowledge known and unforeseen software limitation. Although the problems posed by the limitations were outside of the initial scope, a couple interesting opportunities for future work were discovered. The following discussion will explore a few of the more fascinating and attainable problems.

Run-Time optimisation

An issue with the current algorithm is the length of time it takes to process a route. Ideally the time to generate the GPS artwork should be reduced. In order to achieve this the current algorithm could run less iterations or the algorithm could be optimised. This would be a key area of future development as it is likely it would increase user satisfaction.

Importing Sketches

Currently, the interface the user interacts with to input their drawing is simplistic and does not provide the required tools to design complicated sketches. Furthermore, as the sketch is designed by connecting lines between where the user clicks, it is impossible for the drawing to contain continuous curves. Therefore, the sketch designs are limited. By implementing an option for sketch importation, the user can utilise a pre-drawn shape to maximise the accuracy of the generated GPS art.

Saving Sketches and Routes

By adding user accounts and login functionality, the user will have the option to save GPS art routes previously created and sketches they may wish to use on a different location. This means that the GPS art they generate is not device dependable, as they are able to log in from a different device and access the route. Without the sketch importation functionality, the user may spend a long time designing a sketch. If, for some reason, they wish to reuse the sketch, it may be difficult to replicate the sketch identically. This issue can be fixed by supplying an option to save both the route and sketch within the web application.

Conducting A User Study

Only one trials has been conducted on the validity of the routes generated. Although this test was successful it is essential to conduct further studies the viability of the artwork produced. Ideally this would involve all possible modes of transport, a range of route lengths, and a large variety of sketch shapes and locations. A study like this would help find bugs in the system and provide more information on the accuracy of the calculated routes. A survey could also be conducted to evaluate the features the users would like to see in future development as well as features that need improving.

8.2 Author's Reflection

This project consisted of technical achievement in the field of GIS. Complex issues regarding routing, street networks and comprehensive system evaluation have been addressed. My lack of familiarity with geographical information systems made this project all the more enlightening, as I have learned a great deal throughout the process. Thanks to the university, I have had the opportunity to enhance my academic skills, particularly in time management and perseverance; I feel deeply grateful for the personal growth I have experienced as a result. Although the project has ended, I look forward to improving the algorithm and deploying the website to become publicly available.

Bibliography

References

[1] Wikipedia, "GPS drawing." `http://en.wikipedia.org/w/index.php?title=GPS%20drawing&oldid=1062973042`. [Online; accessed 11-October-2022].

[2] T. Lauriault and J. Wood, "Gps tracings â personal cartographies," *Cartographic Journal, The*, vol. 46, pp. 360–365, 11 2009.

[3] Google, "The artist who crafted a wedding proposal with Google Earth." `https://about.google/stories/gpsart/`. [Online; accessed 10-April-2023].

[4] Y. Takahashi, "MARRY ME." `https://gpsdrawing.info/en/works/marry-me/`. [Online; accessed 10-April-2023].

[5] Guinness World Records, "Largest GPS drawing (individual)." `https://www.guinnessworldrecords.com/world-records/largest-gps-drawing`. [Online; accessed 10-April-2023].

[6] J. Wood, "Traverse Me." `http://www.gpsdrawing.com/maps/traverse-me.html`. [Online; accessed 10-April-2023].

[7] D. E. Warburton, "Health benefits of physical activity: The evidence," *Canadian Medical Association Journal*, vol. 174, no. 6, pp. 801–809, 2006.

[8] R. Guthold, G. A. Stevens, L. M. Riley, and F. C. Bull, "Worldwide trends in insufficient physical activity from 2001 to 2016: A pooled analysis of 358 population-based surveys with 1.9 million participants," *The Lancet Global Health*, vol. 6, no. 10, 2018.

[9] Y. J. Yang, "An overview of current physical activity recommendations in primary care," *Korean Journal of Family Medicine*, vol. 40, no. 3, pp. 135–142, 2019.

[10] G. Boeing, "OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126–139, 2017.

[11] D. G. Kendall, "A survey of the statistical theory of shape," *Statistical Science*, vol. 4, no. 2, pp. 87–120, 1989.

[12] Wikipedia, "Procrustes analysis." `https://en.wikipedia.org/wiki/Procrustes_analysis`, 2023. [Online; accessed 1-May-2023].

[13] P. Perner, "Determining the Similarity Between Two Arbitrary 2-D Shapes and Its Application to Biological Objects," *International Journal of Computer Software Engineering*, vol. 3, 11 2018.

[14] D. K. Rosner, H. Saegusa, J. Friedland, and A. Chambliss, "Walking by Drawing," pp. 397–406, 2015.

[15] University of Washington, "UW mapping app turns art into a sharable walking route." `https://www.washington.edu/news/2015/05/06/uw-mapping-app-turns-art-into-a-sharable-walking-route/`. [Online; accessed 9-October-2022].

[16] "Strava." `https://www.strava.com/`. [Online; accessed 1-May-2023].

[17] Metropolitan Police, "Nuisance behaviour by groups of people." `https://www.met.police.uk/advice/advice-and-information/asb/asb/antisocial-behaviour/nuisance-behaviour-groups-people/`. [Online; accessed 18-April-2023].

[18] Department for Transport, "The Highway Code." `https://www.gov.uk/guidance/the-highway-code/general-rules-techniques-and-advice-for-all-drivers-and-riders-103-to-158#rule149`.

[19] A. Arango, J. Pérez, and B. Poblete, "Hate speech detection is not as easy as you may think," *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019.

[20] Wikipedia, "OpenStreetMap." `http://en.wikipedia.org/w/index.php?title=OpenStreetMap&oldid=1114237081`. [Online; accessed 11-October-2022].

[21] S. Gillies *et al.*, "Shapely: manipulation and analysis of geometric objects." `https://github.com/Toblerity/Shapely`, 2007–.

[22] Wikipedia, "Affine transformation." `https://en.wikipedia.org/wiki/Affine_transformation`. [Online; accessed 22-November-2022].

[23] E. W. Weisstein, "Shear." From MathWorld–A Wolfram Web Resource. `https://mathworld.wolfram.com/Shear.html`. [Online; accessed 24-April-2023].

[24] "Welcome to Flask." `https://flask.palletsprojects.com/en/1.1.x/`. [Online; accessed 4-April-2023].

[25] SISTRIX, "URL length: how long can a URL be?." From MathWorld–A Wolfram Web Resource. `https://www.sistrix.com/ask-sistrix/technical-seo/site-structure/url-length-how-long-can-a-url-be`. [Online; accessed 24-April-2023].

[26] python visualization, "Folium." `https://python-visualization.github.io/folium/`.

[27] Leaflet, "LeafletJS." `https://leafletjs.com`. [Online; accessed 30-April-2023].

[28] T. Kroeger, "GPXpy: GPX file parser and GPS track manipulation library." `https://github.com/tkrajina/gpxpy`, 2018.

[29] Strava, "Uploading Route Files."

[30] Natural Earth, "Cultural vectors, admin 0 - countries [shapefile]." `https://www.naturalearthdata.com/downloads/50m-cultural-vectors/50m-admin-0-countries-2/`. [Online; accessed 1-May-2023].

[31] J. Harvey, "Garmin Connect." `https://connect.garmin.com/modern/course/154875800`. [Online; accessed 17-April-2023].

[32] J. Harvey, "ProjectEdited | 11.5 Running Route on Strava." `https://www.strava.com/routes/3082621732358963146`. [Online; accessed 17-April-2023].

[33] J. Harvey, "Garmin Connect." `https://connect.garmin.com/modern/activity/10922675122`. [Online; accessed 17-April-2023].

[34] J. Harvey, "Star | Run | Strava." `https://www.strava.com/activities/8906449511`. [Online; accessed 17-April-2023].

[35] Strava Support, "Routes on Web." `https://support.strava.com/hc/en-us/articles/216918387-Routes-on-Web#:~:text=be%20found%20here.-,Estimated%20Moving%20Time%20for%20a%20Strava%20Route,last%204%20weeks%20of%20activities.` [Online; accessed 27-April-2023].

[36] "Planet OSM." `https://planet.openstreetmap.org`. [Online; accessed 7-April-2023].

[37] nathanrooy, "TaxiCab." `https://github.com/nathanrooy/taxicab`. [Online; accessed 7-April-2023].

[38] B. Jenny, "Adaptive composite map projections," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2575–2582, 2012.

[39] T. Schauppenlehner *et al.*, "GPS Drawing as a Tool to Examine Spatial Patterns and Relationships," *Rethinking Urban Space*.

[40] M. Modsching, R. Kramer, and K. ten Hagen, "Field trial on gps accuracy in a medium size city: The influence of built-up," in *3rd workshop on positioning, navigation and communication*, vol. 2006, pp. 209–218, 2006.

[41] "Anaconda Software Distribution." `https://docs.anaconda.com/`, 2020.