Final Report Submission

FOR

"ROAD RESURFACE PROJECT"

Submitted to

FACULTY OF IT, MONASH UNIVERSITY

UNDER THE GUIDANCE OF
DR Rasika Amarasiri
2018

Word Count: 3200 (excluding appendices, code, figures)

Abstract

This project presents software that uses data acquired from OpenStreetMaps and VicRoads Open Data to estimate the cost of resurfacing all roads bounded by a user-selected bounding area. The project provides an elegant user friendly GUI.

Keywords

OpenStreetMap, Roads, Resurface, Estimation, Area

Table of Contents

Table of Figures

1.0 Introduction

The project outlined in this report follows a scenario where a local city council requests the development of a software package capable of estimating the cost of resurfacing all roads in a designated 1 km^2 area with the following requirements

- Access to Google maps or another mapping service to select an area.
- Flexibility when selecting an area, i.e. the ability to move around the selected area.
- An elegant and friendly user interface as the application would be used by local council contractors and employees.
- The project needs to function within Victoria, Australia.

Currently, the way that this estimation process is done is through manual labour. The current method is not only cost, it can also cause a disturbance to the road network as certain roads might need to be closed for a short period of time in order to be measured. This project provides a cheaper and easier alternative.

To achieve the requirements and deliver satisfactory and usable software, the project was completed using various technologies. The frontend of the project was developed using PyQt5, HTML, CSS and JavaScript; while the backend of the project was implemented in Python. The implementation style that best fit the constraints involved using OpenStreetMaps (OSM) and VicRoads Opendata as a source of data.

2.0 Background

2.1 Research

During the project planning phase, multiple approaches to sourcing map data were considered, including Image Processing and Machine Learning. This is due to major services, such as Google Maps and Apple Maps not including accurate road widths in their user facing data. In the two services mentioned and many others, roads are displayed at a constant width for convenient presentation; for example, in the Figures 1 and 2 below, there is a large discrepancy between the map and satellite view

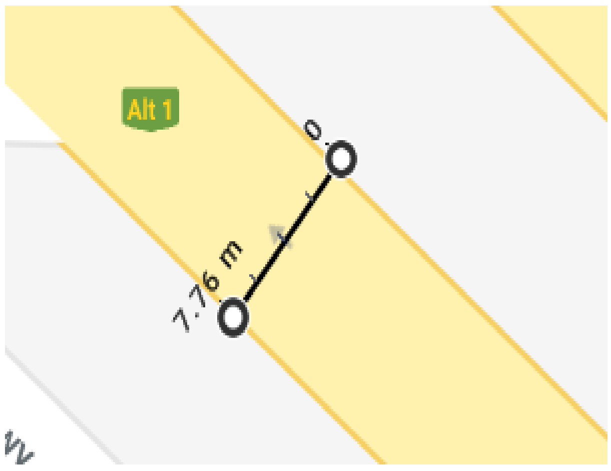


Figure 2 Map view from Google Maps

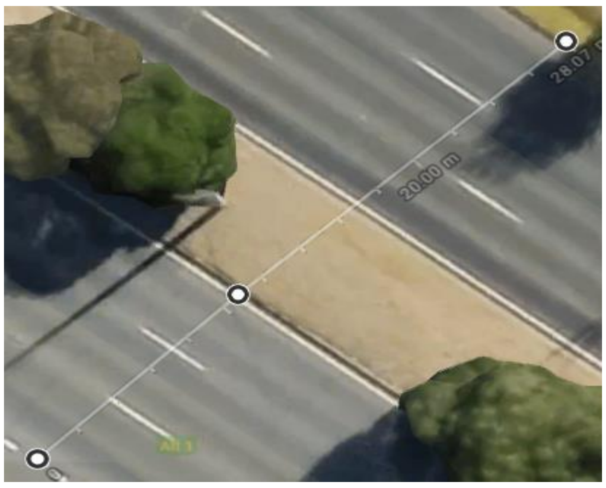


Figure 1 Satellite View from Google Maps

Satellite images could be used to get an accurate reading of the width, as shown in Figure 1. However, when processing a larger area, it becomes much incredibly difficult to extract the road widths from these images. For larger areas, trees, shadows, colours matching those of the road all produce far too much noise to be filtered using simple image processing methodologies. Gecen and Sarp (2008) suggest that that low resolution satellite images make extracting roads near impossible and that in high resolution satellite imagery, we can extract the roads; however, the results would not be accurate enough. Thus image processing alone would not suffice.

Another approach was suggested by [Hu, 2016], experimenting with image filtering and processing alongside machine learning to extract road data and centrelines from high resolution images through various algorithms. The overall strategy discussed proves useful when it comes to extracting road data and proposes a method for shape analysis of the final output; however, to perform such extraction, the project would require high performance computers as well as a large training and validation dataset. Time constraints and the difficulty in sourcing a sufficiently sized public dataset made this approach impractical. Instead, the approach chosen involves processing vector data using the Overpass API, which extracts road data from OpenStreetMaps in the form of "ways" which are saved as straight lines going through nodes. From this data, it is quite easy to extract the length of a road.

Overpass API does not account for road widths; however, using VicRoads OpenData could be used to supplement this initial source, which contains over 70,000 roads within the state of Victoria in the same format Overpass API presents. Where the widths of a road could not be determined by data included in OpenStreetMaps or the VicRoads OpenData set could be estimated based on their classifications, number of lanes and speed limits as listed in "VicRoads Supplement to Austroads Guide to Road Design, Part 3: Geometric Design." [VicRoads, 2016]

2.2 Project Risks

Risk ID	**Risk**	**Mitigation Strategy**	**Probability**

1	Overpass API or Vicroads Opendata go offline	N/A	Low
2	Vicroads Opendata fails to update road data if changed	N/A	Low

The risks listed in the table above have no mitigation strategy as there is no account for them within the scope of the project. Risk #2 is unlikely to happen as the Vicroads Opendata dataset has been receiving updates annually since 2015. [VicRoads Opendata, 2017]

2.3 Resource Requirements

The project requires a strong internet connection to correctly source the required data.

Due to the project being implemented in Python, tools such as PyInstaller can be used to bundle up a single executable with all the required packages. If this option is not taken, the following Python modules are required:

- Overpy
- PyQt5
- Numpy

Instructions on installing these will be provided in the Appendices.

2.4 Project Timeline

The project timeline from the Project Specification Report can be seen in figure 3.

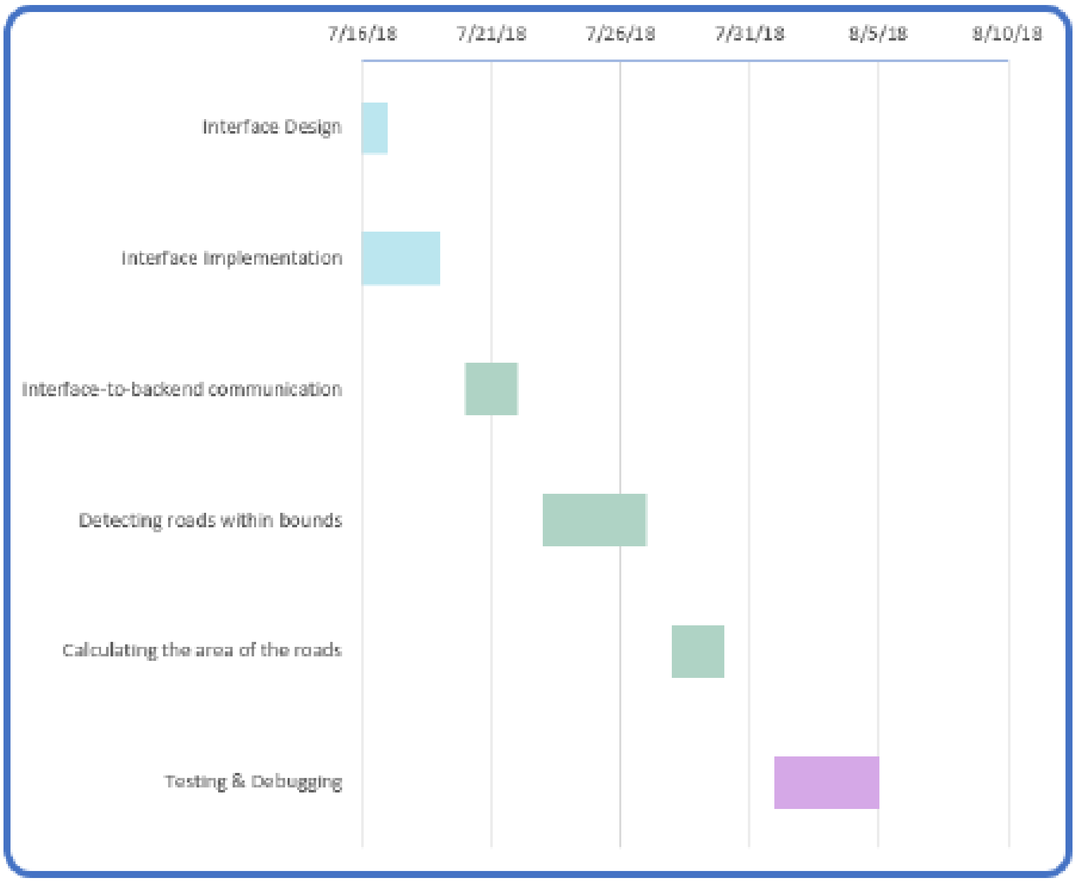


Figure 3 Gantt Chart

This timeline was followed throughout the project; however, the 'Detecting roads within bounds' was achieved sooner than expected, and thus, which freed up more time to work on the interface than originally planned.

3.0 Method

The chosen methodology involves querying OpenStreetMap(OSM) via Overpass API and using VicRoads Opendata to access road information and vector data regarding the roads. The method used by both services involves utilizing a conceptual data model consisting of nodes and ways. Nodes are points with known latitudes and longitudes; while a way is an ordered list representing linear features, such as roads and rivers, between two and two thousand nodes. Thus, when it comes to describing a way, a list of involved nodes is enough information. [OpenStreetMaps, 2018]

When it comes to calculating the area per road, finding the distance between consecutive pair of nodes allows calculation of the length of any road. Due to the OSM data model the line connecting any two nodes is a straight line, thus curvy roads are bound to have more nodes, thus this mode of calculation proves effective when measuring any road. This was implemented using PyQt5, JavaScript, HTML and CSS to create the frontend and Python for the backend

3.1 Internal Design

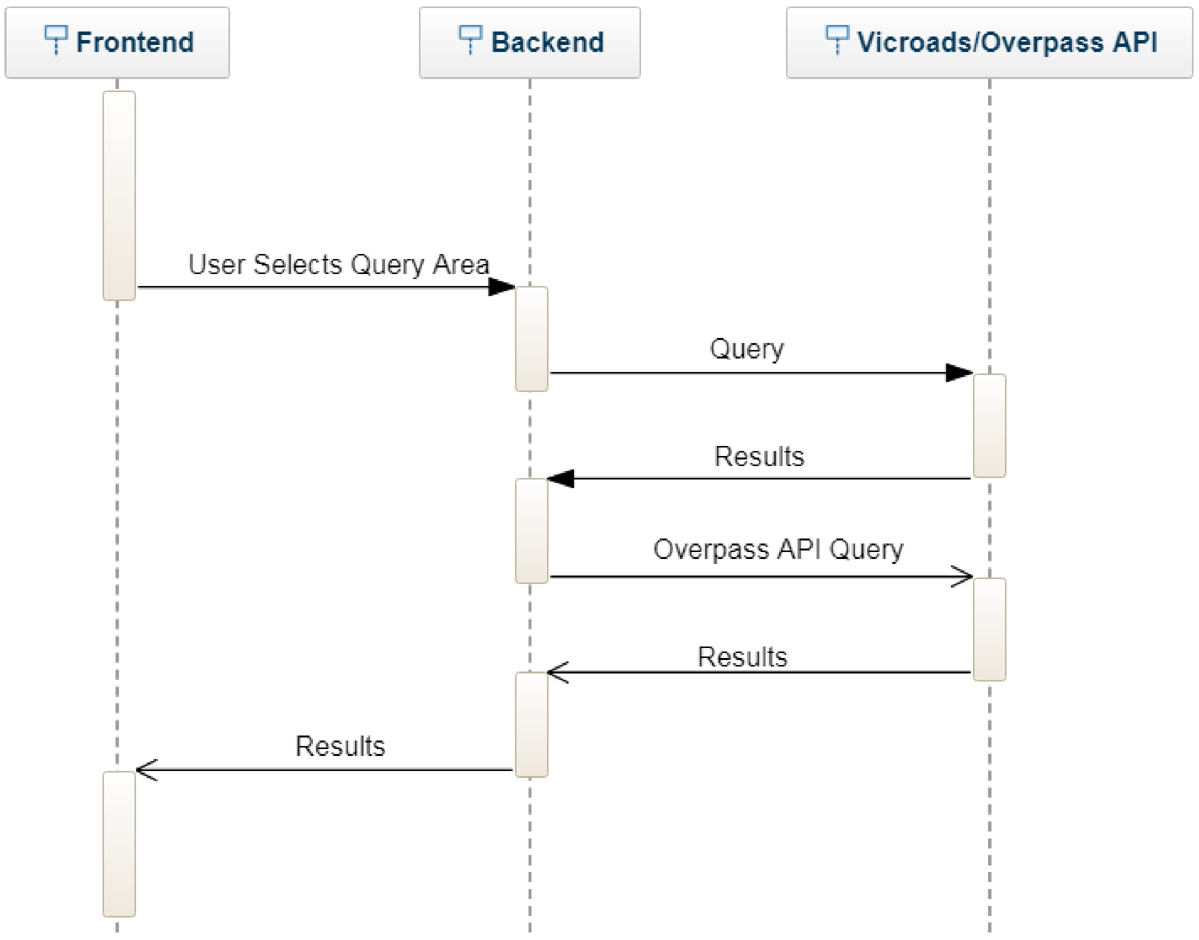


Figure 4 Sequence Diagram of the application

The sequence diagram above shows a typical execution of the program.

3.1.1 Frontend

The frontend has two primary purposes:

- Allow the user to designate an area on a map and deliver the bounding coordinates of the selection to the backend.
- Display the results once all calculations are returned from the backend.

To achieve the two primary goals for the frontend, it uses PyQt5's QtWebEngine to display a browser window. There are two windows, the first contains the map and other options related to area selection. The second display contains a browser display and a PyQt5 button. When the 'Calculate Quote' button is pressed, the bounding area is transferred from the frontend to the backend. The browser display is powered by Leaflet maps, and Leaflet-areaselect. Those two together allow accurate selection of the bounding box.

Chart.js will be used to provide interactive charts to better visualise the data once results are acquired.

3.1.2 Backend

When the boundary comes through from the frontend, it comes in the form of a southwest and a northeast point, which are then rearranged to be left, bottom, right, top latitude and longitude points. As stated in section 2.0 of this report, the software uses the Overpass API and VicRoads OpenData API to extract the roads enclosed in our boundary.

The next step is to query the VicRoads OpenData API for the boundary and load up each result into a class. The way this works will be further explained in the Software Architecture. As each *way* is processed, a filtered version of the way's name and the way's area are stored. The Overpass API query is executed next on the same boundary. Ways that have a matching name with the previously sorted ways is discarded to avoid measuring the same way twice. Once all areas are calculated, the next step is to return the results to the frontend to be displayed.

3.1.3 Communication

The way the backend communicates with the frontend is via PyQt5's command:

runJavaScript(command, callback).

Thus, when a button is pressed, such as the Estimate Quote button, a Javascript command is run on the browser through PyQt, and the response is returned via a callback, ready to be parsed and processed. This same process is used to communicate the results back to the results page.

An example of this:
Backend:

```
self.form_widget.browser.page().runJavaScript("get_center()",
self.update_center)

def update_center(self, string):
    self.center = string.split(",")
    self.center[0] = float(self.center[0])
    self.center[1] = float(self.center[1])
    print(center)
```

Frontend:

```
function get_center() {
    return center_lat+"," + center_long;
}
```

3.2 Software Architecture

The PyQt5 GUI runs through a MainWindow class, which then creates the AreaPage class, this contains everything relevant to selecting the designated area. The AreaPage class contains a 'Calculate Quote' button which when clicked will grab the bounding box coordinates and send it to the backend for calculation and creates the ResultsPage class that displays the results. Class diagrams for all these classes can be seen in the figures below.

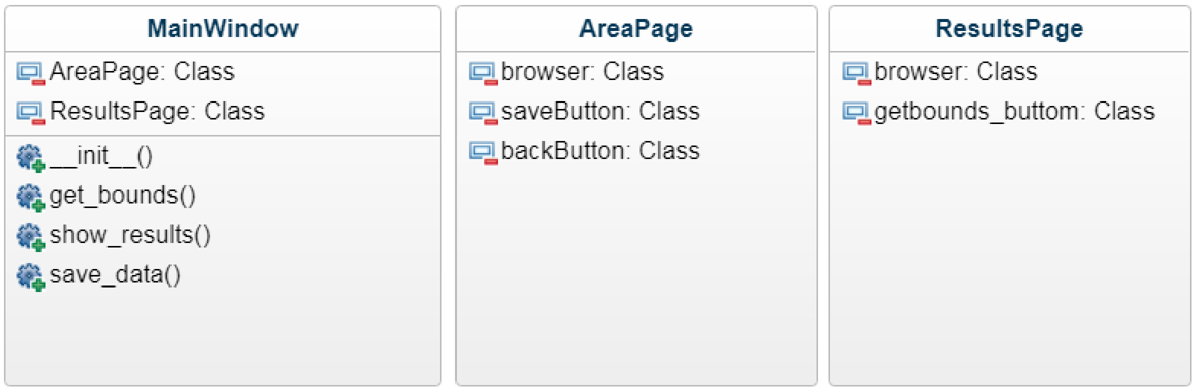


Figure 5 UML diagrams

As discussed in section 3.1.2, ways are extracted from VicRoads Opendata API and Overpass API and loaded up into classes, being vRoad and Road respectively. The distinction must be made between the two services as they both return their data in different styles and thus there would be some differences that can be handled separately in a simpler way than to combine those. Figure 6 shows a UML diagram of both classes.

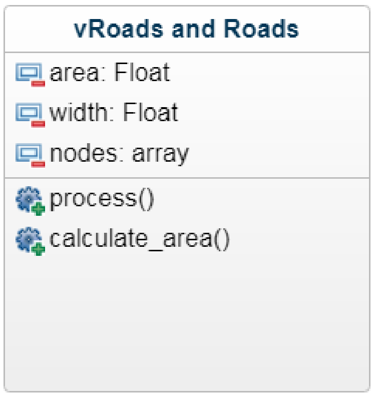


Figure 6 UML diagram

When it comes to querying Overpass API, Overpy provides the ability to run queries with a few commands. The Overpy query will take the form of:

```
way("left, south, right, north") [\"highway\"];" + """
(._;>;);
out body;
```

Meanwhile Vicroads Opendata query is performed by performing an HTML get request within python using the built-in library urllib.request. Once the response from the request is extracted in the form of JSON, it is decoded and relevant information is extracted.

After data is passed onto the road class, the first step is to process the nodes. This is due to both APIs returning the entire way; not just the segment inside the bounding box. Nodes outside the bounding box are removed, except for the first node outside and the first node inside. This done in-order to estimate the intersection of the two in the form of a virtual node. This process will be further explained in section 3.3.1. Once the nodes array is finalised, 'calculate_area()' is called, which then uses the Haversine formula to measure the distance between each node. The distances are summed up and multiplied by the width to return an estimation of the area.

3.3 Algorithms, Key Functions & Formulas

3.3.1 Virtual Nodes

As mentioned in section 3.2, both query API used return an entire way if it intersects our boundary. If we were to take the first node outside the box, then we might gain a lot of extra length. If we were to exclude the first node outside, then there will be distance between the boundary and the first node inside not covered. Thus, this is a very important algorithm.

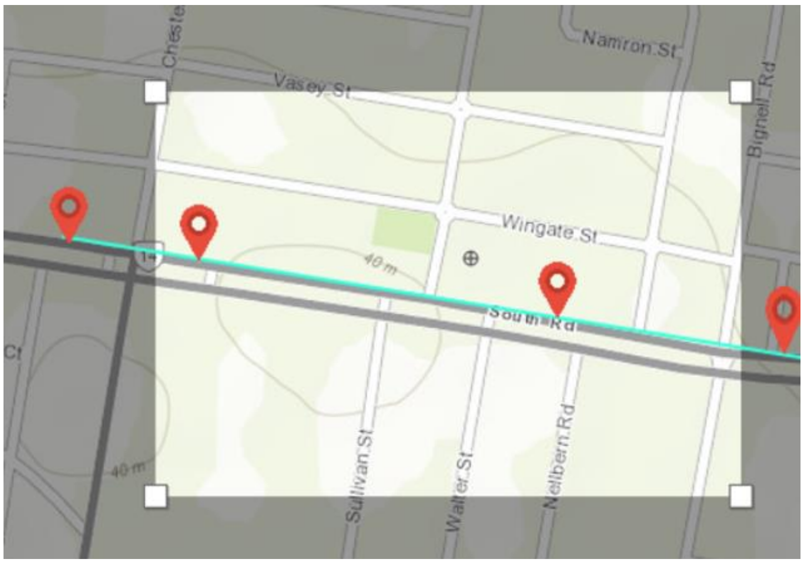


Figure 7 Virtual Node problem example

This is handled by calculating the latitude and longitude of every node between the first node outside and the first node inside the boundary box with an interval of 1 meter. We halt this process until we reach a node within 0.01 degrees of the boundary. Figure 7 and 8 demonstrate the before and after of this algorithm, with the black X showing an example of a virtual node location.

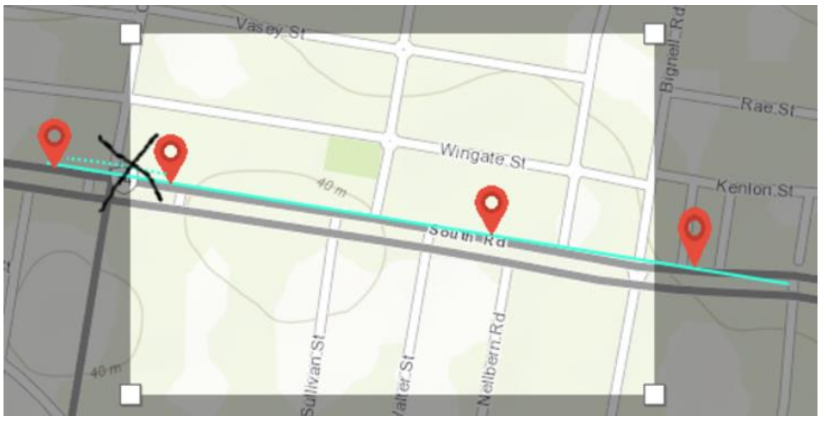


Figure 8 Virtual Node applied

3.3.2 Haversine Formula

Haversine Formula is very important for the functionality of the program, it is used to calculate distances between two points with latitudes and longitudes.

$$d = 2r\arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2-\phi_1}{2}\right)+\cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right)$$

3.3.3 Calculating Zoom Level

Calculating the zoom level is required when it comes to displaying results as the results page is designed to show a static image, grabbed from Google Maps Static API, to give the user an idea of the area they queried. To do so, the boundary extracted is used to calculate what zoom level and centre of map should be used. The formula is shown below:

```
var GLOBE_WIDTH = 256; // a constant in Google's map projection
var west = bounds[1];
var east = bounds[3];
var angle = east - west;
if (angle < 0) {
  angle += 360;
}
var zoom = Math.round(Math.log(600 * 360 / angle / GLOBE_WIDTH) / Math.LN2);
```

3.3.4 Dimensions in Meters

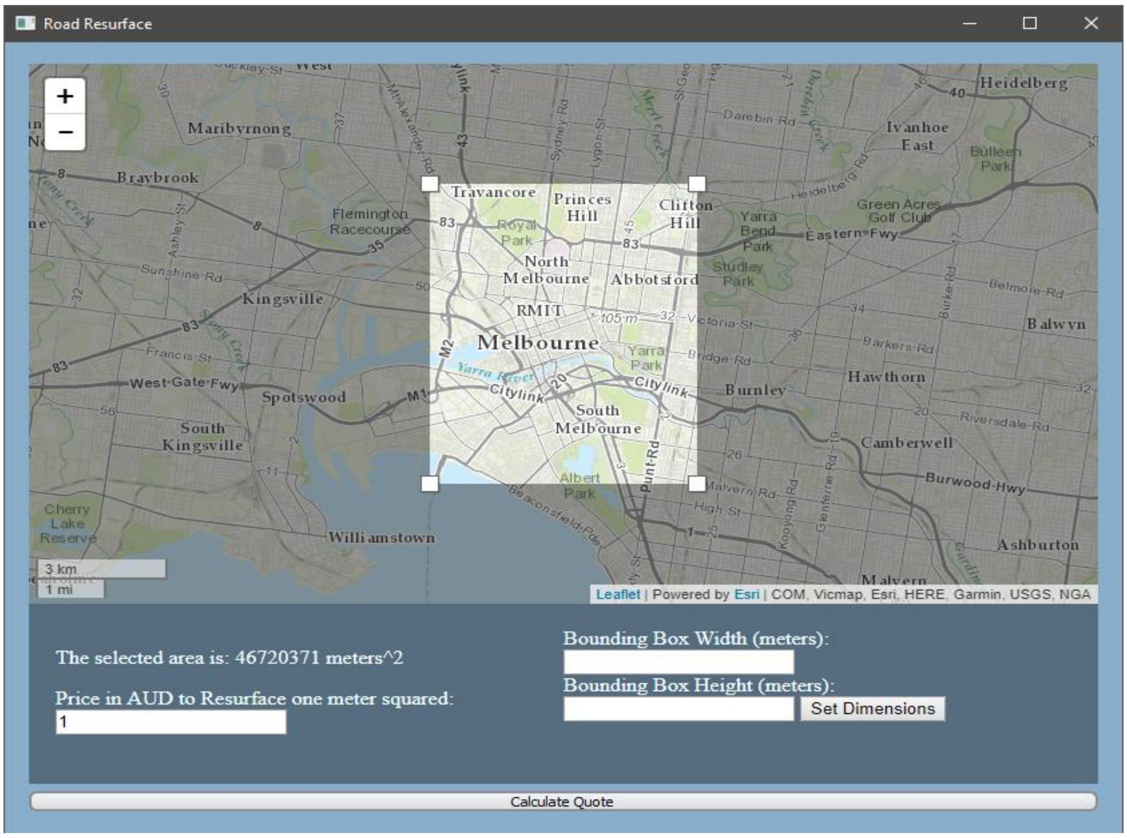
Leaflet-Areaselect provides a way to select area on a map; however it does not support setting the dimensions of an area in meters, therefore we construct a function, setDimensionMeters() which finds the centre of the current map view in latitude and longitude, and then finds the latitude and longitude of a point 100 pixels to the right of it, and another 100 pixels up from it. Haversine's formula is then used to calculate the distance between the centre and each one of those points, and then distance divided by 100 to get the meters per pixel ratio. 100 pixels is used is that on high zoom levels, as using one pixel gives inaccurate results

4.0 Results

4.1 Implementation Outcomes

4.1.1 GUI Implementation

The methodology discussed in section 3.0 was implemented successfully. The UI was originally designed to be a single page application; however, after allowing a few people to test the application and acquiring feedback the decision to split up the UI into two pages was made. The first page focuses on area selection, as shown in figure 9, meanwhile the second page focuses on the results acquired as shown in figure 10.



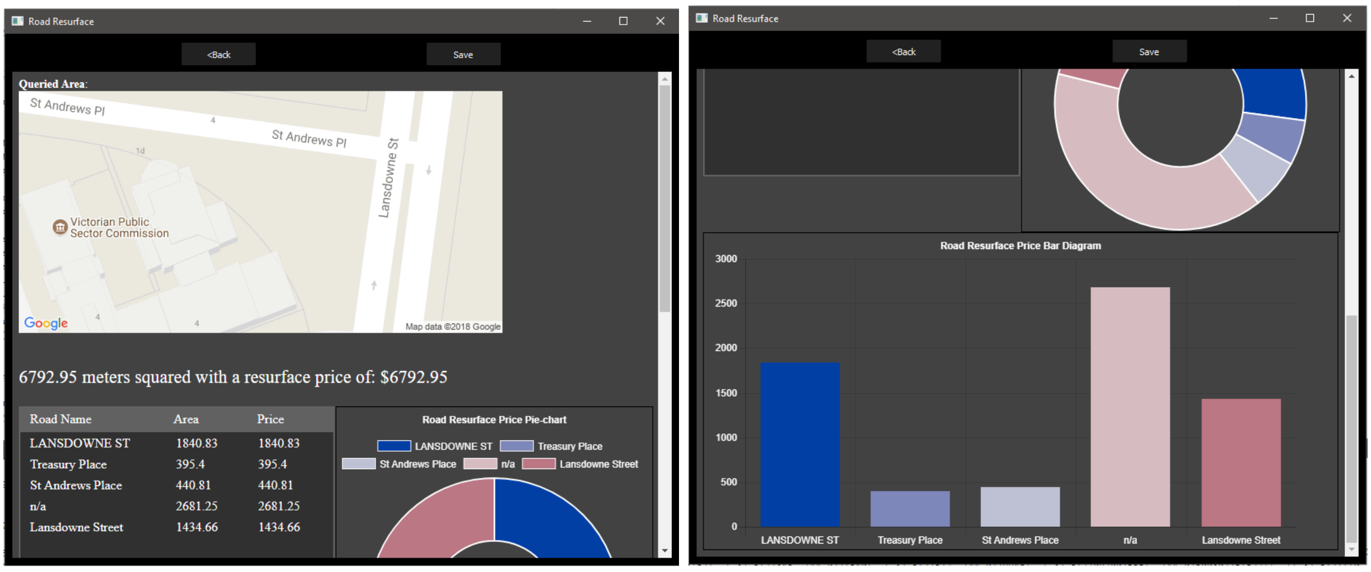
The selected area is: 46720371 meters^2

Price in AUD to Resurface one meter squared: 1

Bounding Box Width (meters):

Bounding Box Height (meters):

Figure 9 Page One of the GUI



6792.95 meters squared with a resurface price of: $6792.95

Road Name	Area	Price
LANSDOWNE ST	1840.83	1840.83
Treasury Place	395.4	395.4
St Andrews Place	440.81	440.81
n/a	2681.25	2681.25
Lansdowne Street	1434.66	1434.66

Figure 10 Results Page of UI

4.1.2 Query Processing

Likewise, the backend was implemented as planned, as the results from various tests, including unit tests, functional tests and others discussed in the Test Report, the results acquired were satisfactory. The virtual node algorithm was implemented to approximate the boundary with low error. The worst case results in only a few meters increase in the total area.

Once the bounding box is extracted from the GUI, the input is parsed and used to query VicRoads Opendata. Resulting roads are immediately calculated, and have their areas stored. All extracted road names are filtered to remove words such as "Hwy, Highway, Freeway, Rd, Road, Fwy" and stored in a separate array. The next phase involves querying Overpass API. All roads extracted via OverPy are cross checked with the filtered names array to ensure that no road is calculated twice.

Once all roads have been processed, the result is returned to the GUI to be displayed.

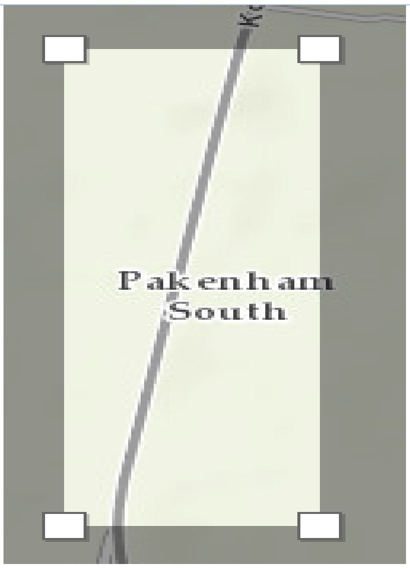
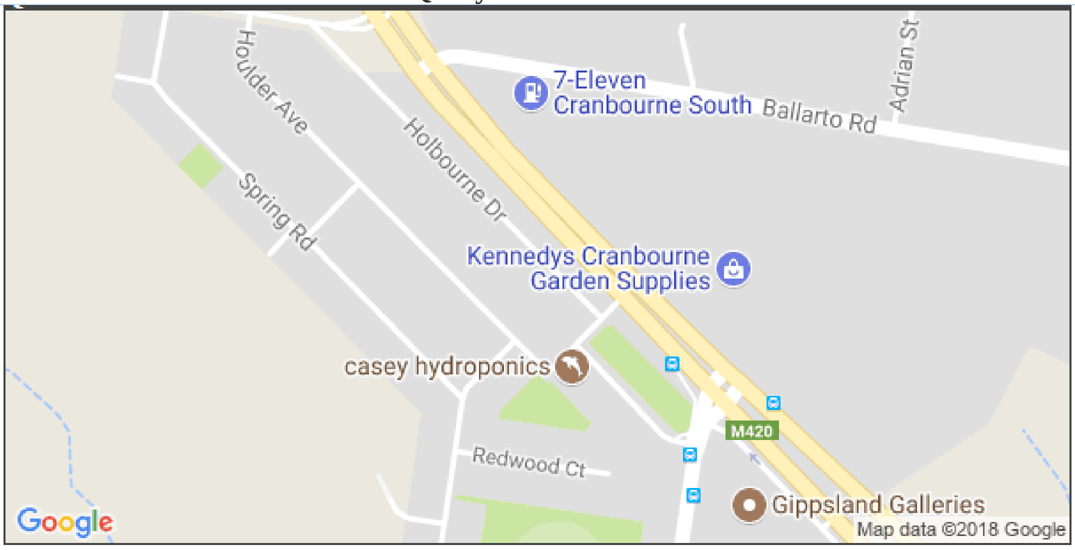
4.1.3 Area Calculations

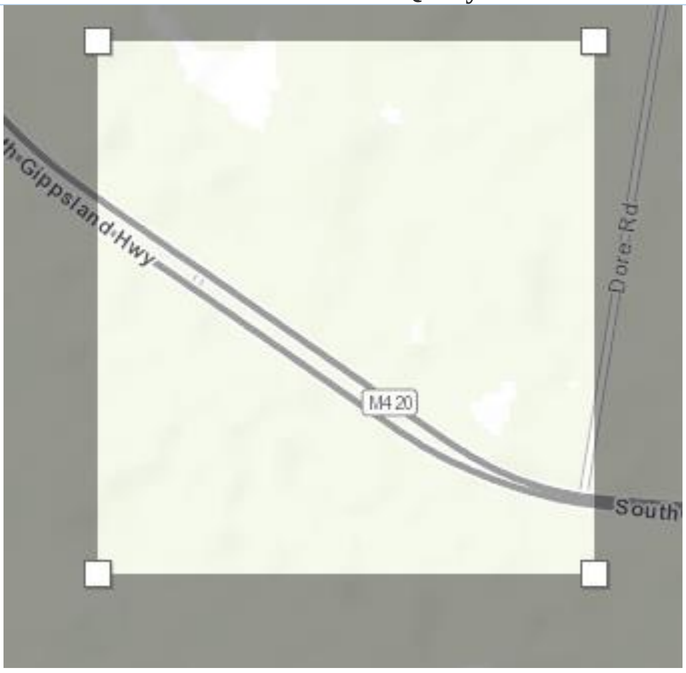
Implementing the calculation phase of VicRoads Opendata API roads was quite simple, once excess nodes are removed and the Virtual Node algorithm from section 3.3.1 is executed if required, the width extracted from the API is multiplied to the total distance between nodes. Meanwhile, results extracted from Overpass API often had no width associated with them, and thus road classifications per in "VicRoads Supplement to Austroads Guide to Road Design, Part 3: Geometric Design" were used to identify the road width.

4.2 Sample Results

For the sake of analysis, smaller areas were queried in this section. This allows easier measuring of project accuracy.

Table 1 Sample Results

Query One	
Query Bounds	 Boundary described by: -38.1372, 145.4873, -38.1291,145.4911 Roads extracted: HEALESVILLE-KOO-WEE-RUP RD
Results:	Total Area: 7700.84 m^2
Query Two	
Query Bounds	 Houlder Ave Holbourne Dr Spring Rd 7-Eleven Cranbourne South Ballarto Rd Adrian St Kennedys Cranbourne Garden Supplies casey hydroponics M420 Redwood Ct Gippsland Galleries Google Map data ©2018 Google Boundary described by: -38.1359, 145.2874, -38.1269, 145.2988 Roads Extracted: 'SOUTH GIPPSLAND HWY', 'n/a', 'Ballarto Road', 'Holbourne Drive', 'The Arcade', 'Craig Road', 'Glendoon Street', 'Redwood Court', 'Houlder Avenue', 'May Road',

	'Spring Road', 'Jennifer Street', 'Staley Street', 'Willoby Street', 'Shaw Road', 'Adrian Street', 'South Gippsland Highway Service Road'
Results:	Total Area: 52663.97 m^2
Query Three	
Query Bounds	 Boundary described by: -38.2079, 145.3601, -38.1990, 145.3707 Roads extracted: 'SOUTH GIPPSLAND HWY', 'Dore Road'
Results:	Total Area: 33028.41 m^2 South Gippsland Hwy - 29294.41 m^2 Dore Road - 3734.35 m^2

4.3 Performance

Measuring the performance of this project proves to be a difficult task as it uses synchronous communications with the Overpass API query server and the VicRoads Opendata API query server. Thus, internet speed is a bottleneck. The table below shows various averaged execution times performed at a download speed of 5 Mbps and an upload speed of 0.53 Mbps.

Table 2 Execution Times

Number of Roads	Avg. Execution Time	Query Area
33	6.34 s	1 km^2
15	3.14 s	1 km^2
2	1.70 s	1 km^2

As the application creates a new class for every extracted road, the time and space complexities to process each single way are O(N) where N is the number of nodes per road. Assuming there are M roads, the space complexity is O(M+N).

5.0 Analysis & Discussion

For the sake of ease of analysis, the queries that will be used for this analysis are queries one and three from Table 1.

Query One – Analysis

Width from Google Maps Satellite view = 10m, Length from Google Maps Satellite view = 773m

$$Area = width\ x\ length = 10\ x\ 773 = 7330\ m^2$$

Application output: 7700 m^2

Query Three – Analysis

Application output:

South Gippsland Hwy - 29294.41 m^2

Dore Road - 3734.35 m^2

Google Maps Satellite View:

South Gippsland Hwy: (1170 * 2) * 13 = 30420 m^2

Dore Road – 851 * 5 = 4255 m^2

As the Google Maps Satellite view extraction of the road lengths shown above is done manually, the length is less likely to be accurate; regardless we can still see that the result extracted by the application is close enough to expected value. In queries one and three, we see that South Gippsland Hwy and HEALESVILLE-KOO-WEE-RUP RD match the manually measured area. From other measurements, such as those outlined in the Test Report, it is evident that roads extracted from VicRoads Opendata API are more likely to be accurate than ones extracted via Overpass API. This is due to having a more accurate measurement of the road width. An example of an inaccurate measurement of the road width is Dore Road, which can be seen in query three. Dore Road is a rural gravel road with a width of 5 meters, consisting of one lane. However, suburban roads extracted from Overpass API are more likely to be accurate, especially if they have their number of lanes noted in the OSM database.

It is hard to measure the accuracy of this application and method due to the lack of a reliable verification method; however, it is evident the program, in most cases, reports reasonable estimations. The project would prove to be more accurate if there was more data to be used or if there was a more accurate method of acquiring the width of queried roads. Regardless, the project still provides an easier way of providing quotations.

6.0 Future Work

Due to time constraints, there were some possible features that were not implemented. Those are:

- Allow exporting to excel spreadsheets. Currently the users can only export to .txt files or copy paste manually.
- Download Victorian road data from OpenStreetMap and apply VicRoad Opendata's road widths to it, allowing the application to be used without an internet connection while removing the internet speed bottleneck. Removing the internet speed bottleneck would make the execution time of large queries very fast as Australian internet is not fast enough currently.
- Extend the scope of the project to all of Australia.
- Adding a search bar to the map to allow easier navigation.

Another approach that could be considered is using Machine Learning via Tensorflow alongside image processing library OpenCV to extract road area from satellite imagery. As previously mentioned in the Background section of this report, this would involve a large dataset of verification and training data; however, if done correctly, the results would be far more accurate and reliable.

7.0 Conclusion

The project aims to deliver software that speeds up the current process of estimating the cost of road resurfacing. Several approaches were considered; however, using vector data, OpenStreetMap data and VicRoads Opendata was deemed the most appropriate given the time and resource constraints of the project. After implementation, it was seen that the project results in reasonable quotations. While the accuracy of the project could be improved, it currently does provide a usable level of accuracy and delivers an easy to use GUI.

Bibliography

Gecen, R., & Sarp, G. (2008). ROAD DETECTION FROM HIGH AND LOW RESOLUTION SATELLITE IMAGES. The International Archives Of The Photogrammetry, Remote Sensing And Spatial Information Sciences., XXXVII, 355-358.

Lv, Ye & Wang, Guofeng & Hu, Xiangyun. (2016). MACHINE LEARNING BASED ROAD DETECTION FROM HIGH RESOLUTION IMAGERY. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XLI-B3. 891-898. 10.5194/isprsarchives-XLI-B3-891-2016.

Overpass API - OpenStreetMap Wiki. (2018). Retrieved from https://wiki.openstreetmap.org/wiki/Overpass_API

Elements – OpenStreetMap Wiki (2018) Retrieved from https://wiki.openstreetmap.org/wiki/Elements

Road Width and Number of Lanes – VicRoads (2018) Retrieved from https://data.vicroads.vic.gov.au/metadata/Road_Width_and_Number_of_Lanes%20%20-%20Open%20Data.html

VicRoads. (2017). VicRoads Supplement to Austroads Guide to Road Design Part 3: Geometric Design (2016) [Ebook]. Victoria. Retrieved from https://www.vicroads.vic.gov.au/~/media/files/technical-documents-new/supplements-to-the-austroads-guide-to-road-design/vicroads-supplement-to-agrd-part-3--geometric-design.pdf

Pyinstaller – Pyinstaller (2018) Retrieved from https://www.pyinstaller.org/

("Leaflet — an open-source JavaScript library for interactive maps", 2018)

Heyman, J. (2016). heyman/leaflet-areaselect. Retrieved from https://github.com/heyman/leaflet-areaselect

Dino Tools (2014). DinoTools/python-overpy Retrieved from https://github.com/DinoTools/python-overpy

Chris Veness, w. (2018). Calculate distance and bearing between two Latitude/Longitude points using haversine formula in JavaScript. Retrieved from https://www.movable-type.co.uk/scripts/latlong.html

Chart JS – chart.js (2018).

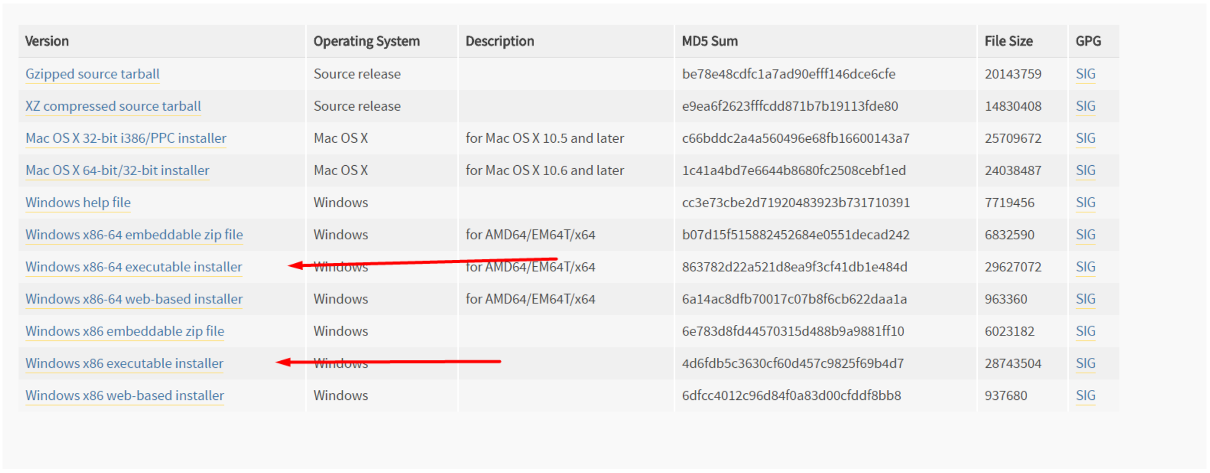
Appendices

Appendix I: Production & Deployment

The project has been tested & compiled on Python 3.6.5 and 3.5.1 on windows platform. Installing either one is required.

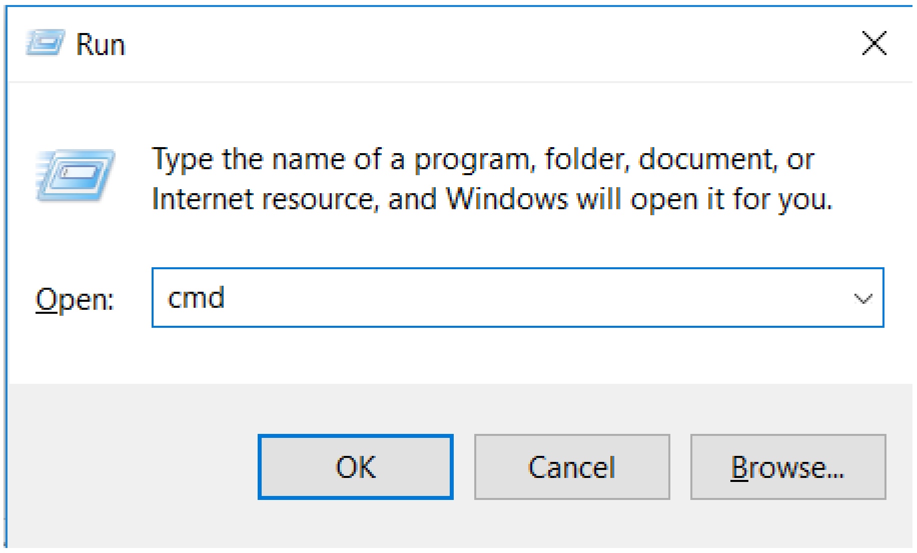
Step 1: If you already have Python installed, you may skip to step 3. Install Python 3.6.5 or Python 3.5.1. Navigate to https://www.python.org/downloads/release/python-365/ or https://www.python.org/downloads/release/python-351/

Step 2: Scroll down to Files and select the executable installer for your platform.



Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		be78e48cdfc1a7ad90efff146dce6cfe	20143759	SIG
XZ compressed source tarball	Source release		e9ea6f2623fffcdd871b7b19113fde80	14830408	SIG
Mac OS X 32-bit i386/PPC installer	Mac OS X	for Mac OS X 10.5 and later	c66bddc2a4a560496e68fb16600143a7	25709672	SIG
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	1c41a4bd7e6644b8680fc2508cebf1ed	24038487	SIG
Windows help file	Windows		cc3e73cbe2d71920483923b731710391	7719456	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	b07d15f515882452684e0551decad242	6832590	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	863782d22a521d8ea9f3cf41db1e484d	29627072	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	6a14ac8dfb70017c07b8f6cb622daa1a	963360	SIG
Windows x86 embeddable zip file	Windows		6e783d8fd44570315d488b9a9881ff10	6023182	SIG
Windows x86 executable installer	Windows		4d6fdb5c3630cf60d457c9825f69b4d7	28743504	SIG
Windows x86 web-based installer	Windows		6dfcc4012c96d84f0a83d00cfddf8bb8	937680	SIG

Step 3: Open command prompt through windows key +R then typing in cmd or simply searching it through the start bar.



Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: cmd

Once inside, type in

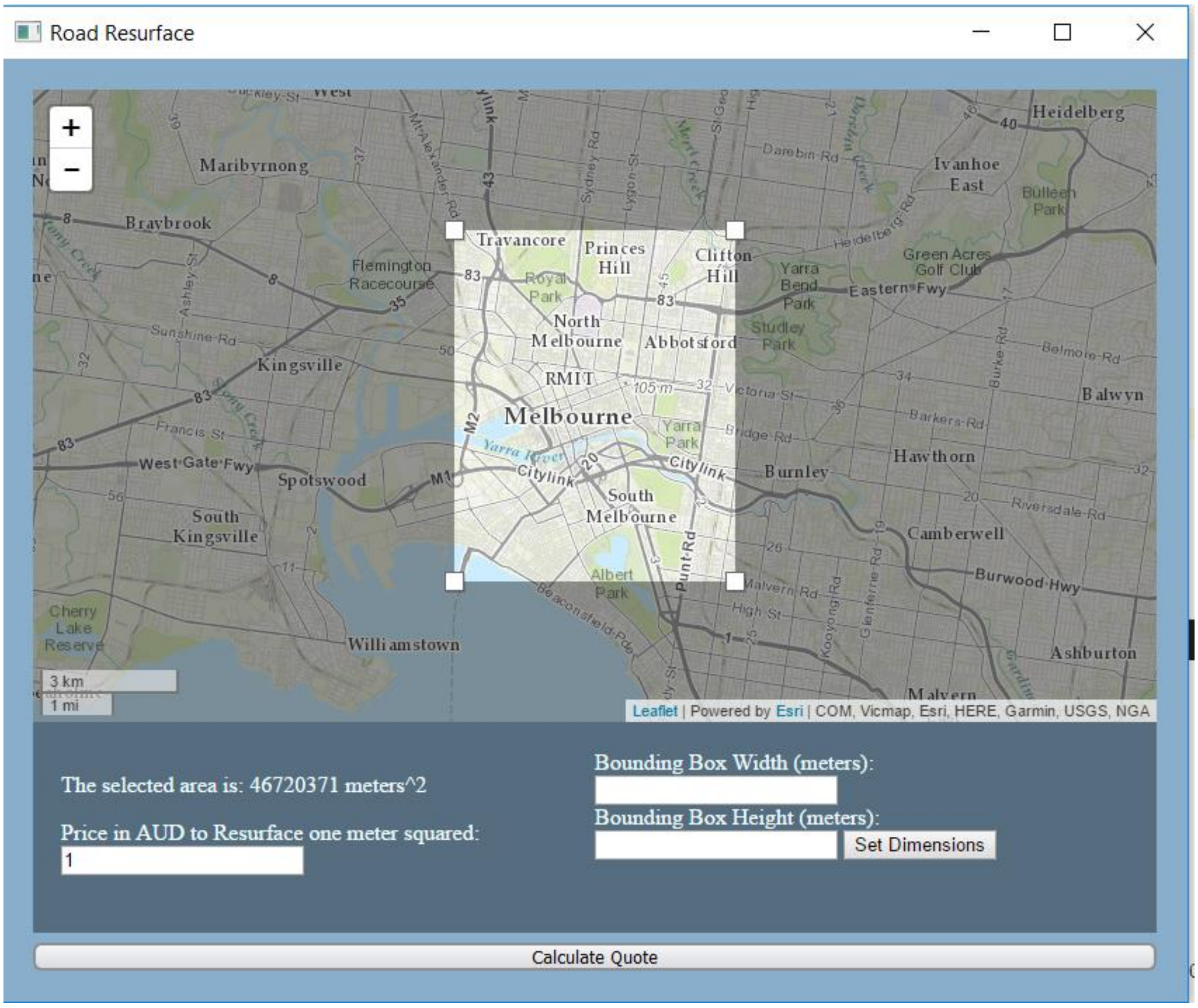
```
pip3 install pyqt5pip3 install overpy

pip3 install overpy

pip3 install numpy
```

If this command does not work, as some windows users have been experiencing issues using pip with the latest versions of python, assuming you have downloaded using Step 1 and Step 2, perform the following commands in cmd:

```
cd %appdata%cd ../Local/Programs/Python/Python36 − 32/Scripts

cd ../Local/Programs/Python/Python36 − 32/Scripts

pip3 install pyqt5pip3 install overpy

pip3 install overpy

pip3 install numpy
```

Step 4: All dependencies should now be installed and ready to be used! Navigate to the project folder and run main.py

Appendix II: User Interface



The selected area is: 46720371 meters^2

Price in AUD to Resurface one meter squared: 1

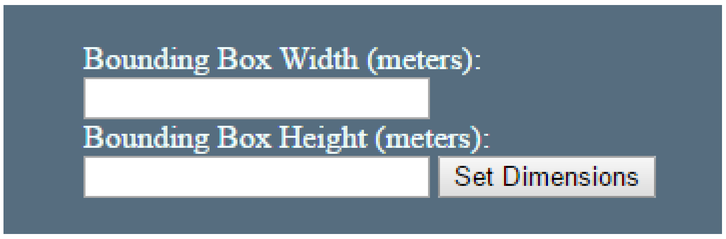
Bounding Box Width (meters):

Bounding Box Height (meters):

When the application is launched, the screen above will be shown.

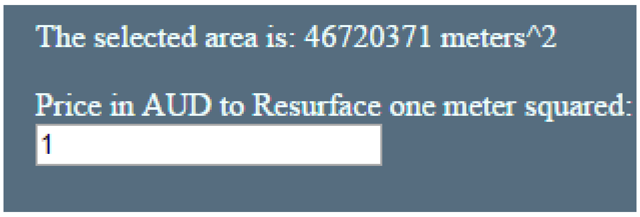
The shaded area is not included, only the bright area (covered by the bounding box) will be queried.

The following fields allow manual selection of bounding box size.


Bounding Box Width (meters):

Bounding Box Height (meters):

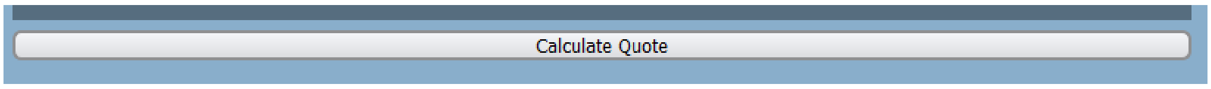
An indication of the query area size is shown, followed by the input for price to resurface one meter squared.


The selected area is: 46720371 meters^2

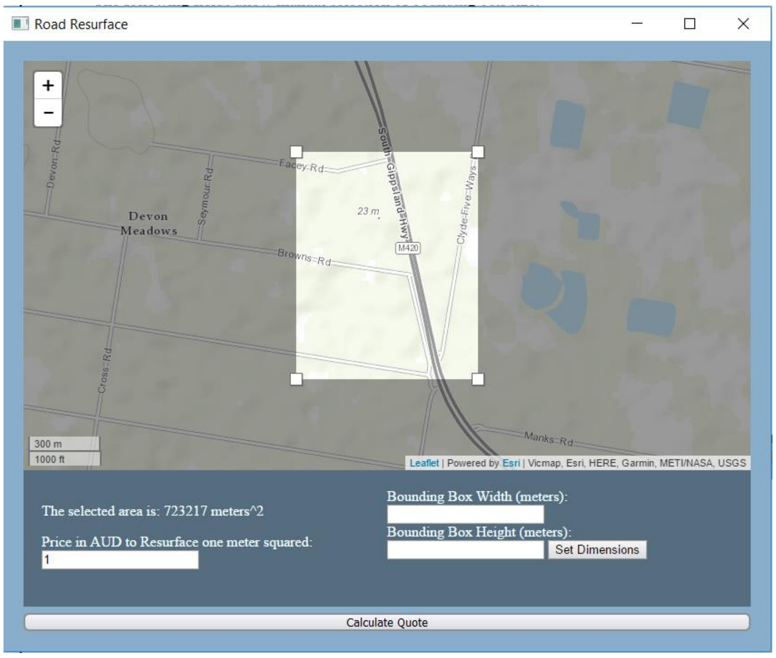
Price in AUD to Resurface one meter squared:

1

Once ready, to query an area, press the Calculate Quote button.



For example, this query:



The selected area is: 723217 meters^2

Price in AUD to Resurface one meter squared:

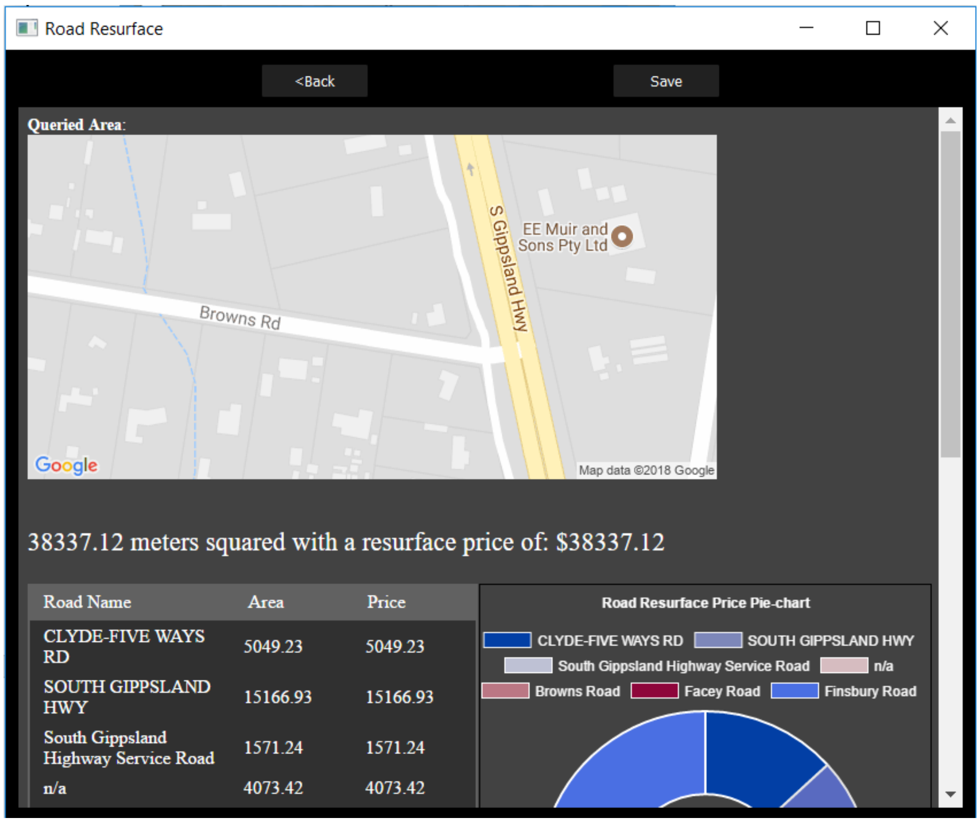
1

Bounding Box Width (meters):

Bounding Box Height (meters):

Once the button is pressed, it will take some time to query and then the results page will be shown.

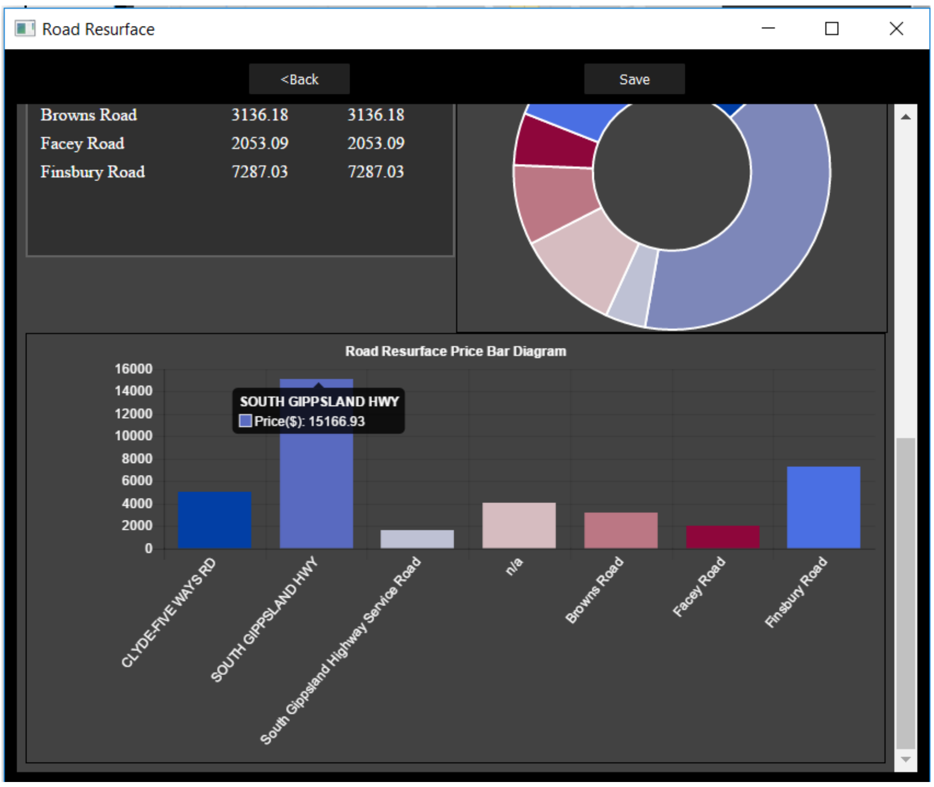
The results page will be shown next:



38337.12 meters squared with a resurface price of: $38337.12

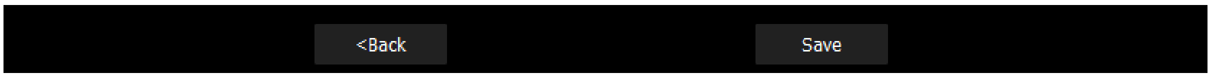
Road Name	Area	Price
CLYDE-FIVE WAYS RD	5049.23	5049.23
SOUTH GIPPSLAND HWY	15166.93	15166.93
South Gippsland Highway Service Road	1571.24	1571.24
n/a	4073.42	4073.42

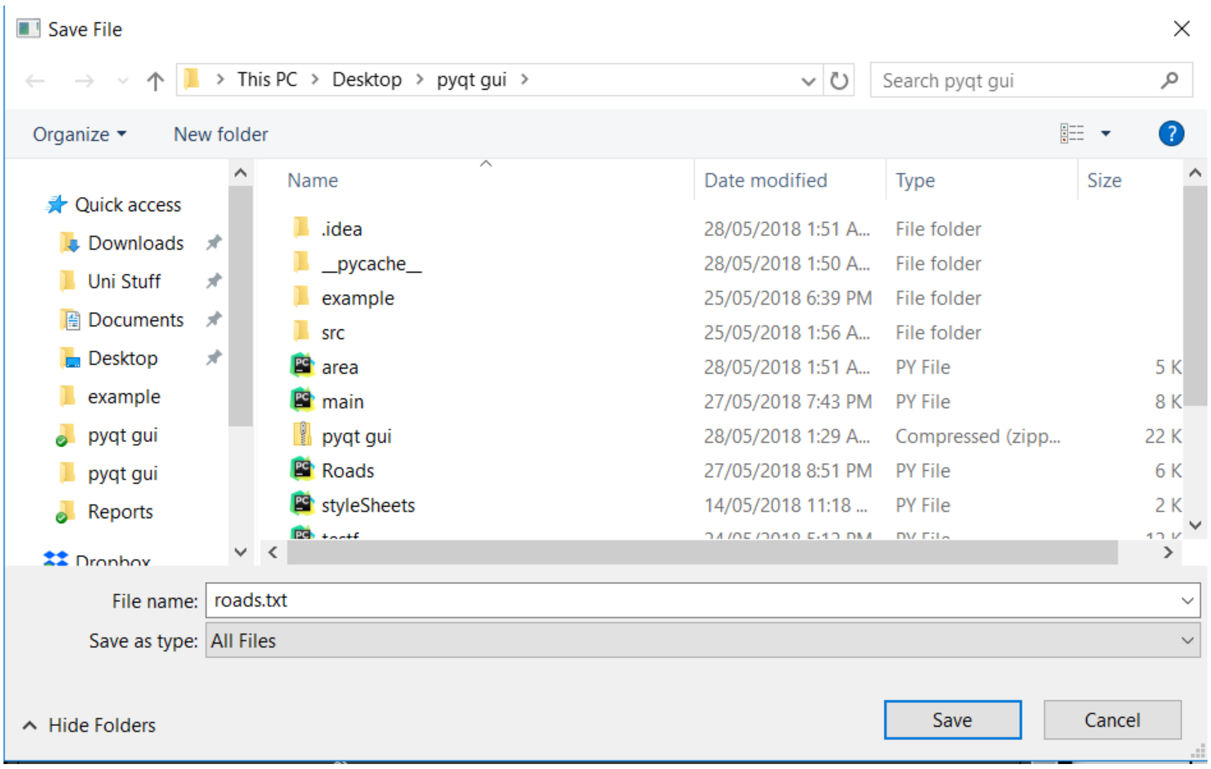
The graphs are interactive:



Browns Road	3136.18	3136.18
Facey Road	2053.09	2053.09
Finsbury Road	7287.03	7287.03

The back button allows you to go back to the selection screen, meanwhile the save button allows you to save the current data into a .txt file.





Name	Date modified	Type	Size
.idea	28/05/2018 1:51 A...	File folder	
__pycache__	28/05/2018 1:50 A...	File folder	
example	25/05/2018 6:39 PM	File folder	
src	25/05/2018 1:56 A...	File folder	
area	28/05/2018 1:51 A...	PY File	5 K
main	27/05/2018 7:43 PM	PY File	8 K
pyqt gui	28/05/2018 1:29 A...	Compressed (zipp...	22 K
Roads	27/05/2018 8:51 PM	PY File	6 K
styleSheets	14/05/2018 11:18 ...	PY File	2 K

File name: roads.txt

Save as type: All Files

Make sure the file extension is there, then click save.

Appendix III: Testing Features

The program includes a unit testing script that is automatically launched when the program is initialised. The name of the file is unit_tests.py, which performs automatic tests of certain functions per the Testing Report. When the application is launched “All tests passed!” should be printed.