Abstract

Keywords: Road Area Calculation, Road Recognition, Satellite

1.0 Introduction

2.0 Background

3.0 Method

3.1 Methodology

Since this project was not overly complicated the methodology used also shows a degree of simplicity. The initial Stages of research were dedicated to finding broad methods of solving this problem, including using research papers, articles and other resources to find people who had solved similar issues. Once I had discovered that there were only two broad techniques one could use I started going deeper into those techniques and created a list of Advantages and Disadvantages for each.

Once I had that list I choose the option I felt best suited for this project, namely using image recognition and counting pixels, and started more rigorous research on that technique. Very quickly I discovered that I could avoid using real satellite imagery and instead use already nicely formatted images thanks to google, so I concentrated my research on issues that would arise for that particular method, namely ways to deal with overlapping roads, changes in elevation and pathways counting as roads.

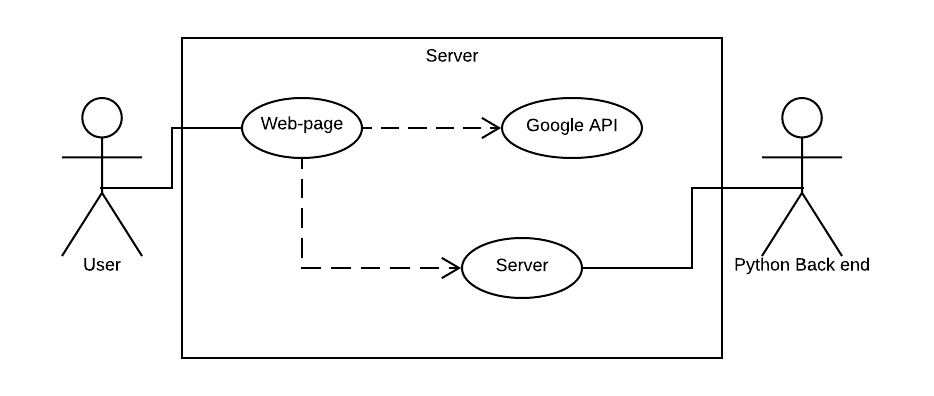
That research done I started on the back-end of the project. Once that was completed I went and made the front end, a process that required a little further research due to my complete lack of experience in the area of GUI development.

3.2 Internal Design

This project has two main parts that work together to create the finished project, The Python backend and the Node.js hosted user Interface. The user interface has a Google map with a movable pin in it [4] , as well as several text boxes. The project works by using a HTML form to POST the latitude and longitude of the pin back to the Node.js server [3] , where using the spawn method will run the python back end with the latitude and longitude as inputs.

Once the Python project has the latitude and longitude from the server, it makes a request to the Google static Maps API [1] for a map centred on those coordinates. This map is in terrain view, and has all detail but roads removed, with each type of road coloured in a different colour. Red, green or blue. Once this map is saved to the hard drive the program will then open it using the Python Image Library [2], and iterate across a band of pixels in the middle (the picture is longer than it is wide to avoid the google watermark) and tallies each colour and then returns the totals of each colour to the Server.

Once the Server has the information, it sends them to the web-page, where they are displayed along with the sum of total road area.



Use Case Diagram [Fig. 3.2]

3.3 Software Architecture

The architecture of this project is not very complicated at all, and since python is not an Object-Oriented Language there is no Classes that need discussion. The Three Python Functions in this project are Get\_Map\_Image(Location), which takes a string of the latitude and longitude concatenated with a comma and saves the google static map to the parent folder, Count\_Pixels(Image), which takes the image saved above and returns the pixel count for each colour to the Standard Output, and Area\_Caculation, which is simply a wrapper that runs the other two functions.

On the Server Side it simply gets the latitude and longitude of the pin, and sends them to the python program, waits till the information is returned and then sends them back to the webpage where they are displayed.

4.0 Results

4.1 General Results  
The project took in a latitude and longitude and returned 4 values, the total area of three different kinds of roads, as well as the total surface area of roads. It was to do this in <10 seconds and with a 10% accuracy, how well it met these goals will be discussed in the following sections but overall it managed to reach the time goal admirably but due to difficulties in measuring accuracy only probably reached the 10% goal.

4.2 Statistics

4.2.1 Accuracy

Measuring the accuracy of this project was complicated as there is no trusted external source by which to verify the results, this means the only way to check accuracy is to either compare with others doing the same project or to calculate by hand. Since calculating by have is either going to be a huge amount of work (Manually measuring) or have many of the same inaccuracy’s as my own project I decided to take images with clear different scales of roads and ensure that the results confirm the ordering, as well as spots centred on the ocean and custom images of all ‘roads’. Having done this and have the results returned as expected it can be said that the program is relatively accurate, but putting an exact figure on the accuracy is beyond the scope of this project.

4.2.2 Speed

The speed of the project is mainly dependant on the request of the image from the google static API, something that will be discussed further in section 4.4.1, with the rest of the operation of the project taking between 1 and 3 seconds depending on the speed of the device its run on and the browser.

4.3 Features

4.3.1 Inputs

The program can take as inputs either the location of a pin of a google map or a individually entered Latitude and longitude into a text field in the UI. These latitudes and longitudes will be as accurate as the Google API allows, which is to 12 decimal places [1]. This provides an accuracy of   
1.118e-10 km (Within Victoria, due to the shape of the earth the level of accuracy changes depending on where you are) or 0.00001118cm. A level of error that is entirely insignificant compared the size of the area we are dealing with and the inaccuracy’s introduced in other areas of the project

4.3.2 Outputs

The outputs of this project are 4 values of the 3 different kinds of roads, these are only calculated to 0 decimal places as the inaccuracy’s of calculating the value of the area are such that any greater accuracy would be fascial. In fact even displaying them to 0 decimal places is more accurate than the program can calculate, but since it is difficult to be sure of the exact accuracy of the project we had to choose a level of accuracy to display, and integer numbers was the most aesthetically pleasing

4.4 Performance

4.4.1 Time

The Time performance of the project was well under the initial specification, with all the calculation times coming in under 2 seconds. The major source of this time was the initial request [Fig 4.2.2]. As you can see from the graph it appears that Google takes longer to return requests that have a greater density of roads, almost linearly.

4.4.2 Space

Because the program does not have any pixel data on the space between roads the size of the image varies with the density of roads (See Fig 4.4.2). This fact couple with the files of the project itself means that the total space requirements for this project range between 1.7 and 2mb

5.0 Analysis and Discussion

5.1 Tests/Data Gathering Techniques

Most of the data gathered was from manually running the code and inserting outputs to the console of relevant information, both on the server side and the backend. A unforeseen complication while recording the results was that the internet connection used for the first half of the results was unavailable for use during the recording of the seconds half, so the speed of the internet differed, which could have caused inaccuracies of the results. Fortunately, the second internet was faster, and since the results showed that the denser images took longer anyway, it means that all the change in internet did was possibly understate the increase in time. Since the direction of the graph is all that matters it did not particularly matter that the result was understated.

The other technique used for gathering data was to cooperate with others doing the same project but with a different program, then to test our programs on the same values, and if we were in 10% of each other this was considered a sign that our programs were accuracy within 10%. This had a problem in that we had both used the same basic design for the project so this would only illustrate errors that arose due to our different execution of the idea, and not general errors inherent to the approach.

5.2 Discussion

The biggest issue with testing this project was the accuracy, as previously discussed. Any discussion of the results will have to account for the great difficulty there was in calculating the total area of road using a method other than your original project. The most accurate way to verify would be to physically survey the area, but this is too expensive and would take to long for it to be viable, the only other options are to try and find alternative methods to calculate the road area, which would basically result in doing the entire project a second time, which for obvious reasons is also not viable. Since we could not use any of these options we were left with the ability to calculate the potential error in our execution, but not the error inherent in the method.

|  |  |  |
| --- | --- | --- |
| First Request  (Secs) | Average Time  (Secs) | % Difference |
| 0.765842438 | 0.561702728 | 73.34442446 |
| 0.714726925 | 0.574992609 | 80.44927216 |
| 0.714412451 | 0.557584715 | 78.04801194 |
| 1.062627792 | 0.766516638 | 72.13406644 |
| 1.186824083 | 0.677441359 | 57.08018299 |

When it came to the size and speed results we could get a much greater degree of accuracy, even if there was a large variance in results. The speed has a large amount of variance, and due to limited resources, I could not test the speed on any internet connection faster than 3G, but there is a large possibility that faster machines and faster internet could decrease the time to run this program by a significant amount. Something worth noting is that for some reason the first request at a set of coordinates was always around 30% longer then any other request. I did not discover why this is and must conclude that it is simply a peculiarity of the way the google static API treats requests, possibly due to it caching the last returned image.

5.3 Conclusion

In Conclusion the results recorded by this project support the main hypothesis of this report, that it is possible to calculate the total surface area of a square kilometre of roads in Victoria within 10 seconds and with 10% accuracy, as discussed, the accuracy might not be within 10%, but it probably is. But with an average time of <3 seconds with a slow internet connection it defiantly proves the time requirement.

6.0 Future Work

This project was very much simply an entry level project on the issue of road calculation, and as such there is a lot of further research that could be done that was beyond the scope of this project, both in simply extensions of the idea as well as further explorations of concepts simply discovered during the project.

In terms of extensions there is a lot of further validation and calculations that could potentially be done from the image to increase the accuracy, such as an algorithm to calculate overlapping roads, that is to calculate the area or roads that cannot be seen from above due to being overlapped by other roads. Or a way to factor in the different elevation of roads into your calculations, not to mention a way of excluding elements that google treats as roads but are not, such as pathways and tracks. There is also the possibility that calculating from the direct satellite image, rather than the already modified google image, would return a better result, or at least provide a separate value to compare to to increase validation. This seems like it would unnecessarily elongate the calculation time of the project but it is worth further research to work out the viability.

When it comes to the concepts brought up by this project, it is mainly to do with image recognition or neural networks as approaches to solve this problem, creating a neural network able to recognise roads was beyond the scope of the project, but if it was implemented it would be a much more general solution than the one provided in this project, as well as have much better real-world applications, as it would not require a pre-formatted image to tell were roads are.

7.0 Conclusion

In Conclusion although there are several methods for calculating the area of roads, with varying degrees of accuracy and necessary calculations, using the google static API to pre-format the image was both the fastest in terms of computation as well as the smallest in terms of storage, while this method introduces several inaccuracy’s due to the nature of converting a 3D road system into a 2D image they are within the allowed error margins of the project and further steps could be taken to reduce those inaccuracy’s of more time and resources were to be dedicated to this project.

The lack of recorded research on this topic made it difficult to start, and what research there was on road recognition was almost universally from a street-view perspective, generally to assist with self-driving cars. This lack of research would suggest there are not many real-world applications for a program that would do this, and the few comments I found from other people doing similar work would suggest that using the approach used in this project is almost entirely pointless as any situation where you really need to recognise roads from orbit you will not have the Google Static API to format the image for you, as it is likely that the roads will either be brand-new or un-official.

However, while this solution may not have many real-world applications, it was certainly the best suited to fit the project specification, and while it may not have been useful, in the course of creating it, several ideas that may have real world applications were bought up.

Bibliography

[1]https://developers.google.com/maps/documentation/maps-static/intro

[2] <https://pillow.readthedocs.io/en/5.1.x/>

[3] <https://expressjs.com/en/4x/api.html>

[4] <https://developers.google.com/maps/documentation/javascript/tutorial>

Appendices