THE UNIVERSITY OF NEW SOUTH WALES

SCHOOL OF ELECTRICAL ENGINEERING
AND TELECOMMUNICATIONS

Haze Watch - Data Modelling and Visualisation using Google Maps

by

Amanda Chow

Thesis submitted as part of the requirements for the
Bachelor of Engineering (Electrical Engineering)

Submitted: 22 October 2010 Student ID: 3210263

Supervisor: Dr. Vijay Sivaraman Topic ID: VR27

Thesis title: Haze Watch – Data Modelling and Visualisation using Google Maps **Topic number: VR27**

Student Name: Jing Ting Amanda Chow **Student ID: 3210263**

A. **Problem statement**

In recent years, there has been a growing awareness regarding air pollution and the need for proper monitoring and forecasting of urban air quality around the world. Air quality has worsened in many developing countries, with the escalating number of motor vehicles and movement towards wide-scale industrialization. This leads to increased health risks, particularly for cardiovascular and respiratory illnesses such as asthma and lung cancer. Hence there is an increasing need for people to start monitoring their exposure to air pollution, and we will need a data analysis and visualisation system to demonstrate our everyday exposure.

B. **Objective**

Review relevant literature about data interpolation techniques (IDW and kriging). Build a fast and efficient web application that has a friendly and easy to understand user interface. Supply accurate and reliable air pollution estimates using selected data interpolation models. provide dynamically constructed real-time graphical visualisations based on collected and interpolated air pollution data

C. **My solution**

Google Maps Javascript API and AJAX requests for user interface
PHP script for inverse distance weighting algorithm
CGI script and R language for ordinary kriging algorithm
CGI script and gnuplot language for image generation

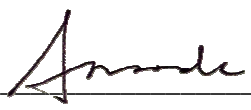
D. **Contributions** (at most one per line, most important first)

Coherent web application that gathers separate functionalities into one
Creating CGI script and R script for kriging algorithm
Creating CGI script and gnuplot parameters for image generation
Creating PHP script for inverse distance weighting

E. **Suggestions for future work**

Field tests with actual pollution sensor and compare with estimation
Implement temporal model, checking dispersion models

While I may have benefited from discussion with other people, I certify that this thesis is entirely my own work, except where appropriately documented acknowledgements are included.

Signature:  Date: _21_ / _10_ / _2010

Thesis Pointers

List relevant page numbers in the column on the left. Be precise and selective: Don't list all pages of your thesis!

7	Problem Statement
7	Objective

Theory (up to 5 most relevant ideas)

12	Ordinary kriging
11	Inverse distance weighting
16	AJAX requests and responses
28	Cross-validation

Method of solution (up to 5 most relevant points)

16	Google Maps and AJAX
18	User interface
22	Kriging implementation
22	Inverse distance weighting
24	Image generation

Contributions (most important first)

15	Coherent web application that gathers separate functionalities into one
22	Creating CGI script and R script for kriging algorithm
24	Creating CGI script and gnuplot parameters for image generation
22	Creating PHP script for inverse distance weighting
18	Javascript for displaying map and markers

My work

15	System block diagrams/algorithms/equations solved
26	Description of assessment criteria used
15-25	Description of procedure (e.g. for experiments)

Results

26-38	Succinct presentation of results
26-38	Analysis
26-38	Significance of results

Conclusion

39	Statement of whether the outcomes met the objectives
39	Suggestions for future research

Literature: (up to 5 most important references)

12	[23] I. Clarke, 2010
13	[26] Y. Zhukov, 2010
22	[31] E. Pebesma, 2001
24	[34] gnuplot, 2010
16	[27] Google, 2010

Acknowledgements

My deepest thanks go to my supervisor, Vijay Sivaraman, whose enthusiasm and vision for the Haze Watch project helped to encourage and guide me through the entire data modelling and visualisation system set-up. I am also grateful to my fellow members of the project, James Carrapetta and Nik Youdale, whose collaborative efforts and ideas acted as inspirations.

I am indebted to the many experts of the Google Maps Javascript API v3 group and the R-sig-geo (R Special Interest Group on using Geographical data and Mapping), who dedicate their time online in answering the questions of others. I would like to thank my friends who happily offered their alternative opinions and viewpoints, and lastly my family, who helped support my efforts throughout this thesis.

Abstract

Air quality has worsened in many urban centres around the world, which leads to increasing health risks. Haze Watch is the start of closer monitoring and analysis of the air quality in Sydney by means of mobile wireless pollution sensors and data visualisation system; giving people the chance to evaluate their personal exposure to air pollution. This thesis documents the development and analysis of a web-based application to store, model and represent data using Google Maps and other integrated systems. We cover similar geographic information systems and relevant background theory in this report and will discuss how to implement important features and ways to assess the application's reliability, before suggesting further improvements that could be made.

Abbreviations

AJAX	Web development techniques that enhance interactivity of web applications
API	Application interface implemented to enable easy interaction between software
CGI	Common Gateway Interface, protocol for calling external programs on a server
CO	Carbon monoxide, poisonous gas
CSS	Cascading Style Sheets, describes look and formatting of a web page
GIS	Geographical Information System, used to analyse geo-referenced data
HTML	Hypertext Markup Language, markup language for web pages
IDW	Inverse distance weighting, a data interpolation method
NO_2	Nitrogen dioxide, poisonous gas and common air pollutant
NRMSE	Normalised RMSE, ratio of standard deviation to RMSE value
O_3	Ozone, air pollutant with harmful effects to humans
PHP	PHP: Hypertext Preprocessor, a general-purpose scripting language
PM10	Particulate matter suspended in air, smaller than 10 micrometers in size
PM2.5	Particulate matter suspended in air, smaller than 2.5 micrometers in size
SO_2	Sulphur dioxide, common air pollutant and precursor to sulphuric acid
RMSE	Root mean square error, measures difference between predicted and actual values
UI	Interface between user and machine
URL	Specifies where identified resource is located and how to retrieve it
XML	Extensible Markup Language; set of rules to encode documents

Contents

Introduction 7

Background 8

2.1 Haze Watch 8

2.1.1 Objectives of the project 8

2.1.2 Project topology 8

2.2 Similar products 10

2.3 Data interpolation techniques 11

2.3.1 Inverse distance weighting interpolation 11

2.3.2 Kriging interpolation 12

Implementation 15

3.1 System overview 15

3.2 Client-side implementation 17

3.2.1 Google Maps and AJAX 17

3.2.2 User interface 18

3.3 Server-side implementation 21

3.3.1 Database access 21

3.3.2 Data interpolation models 22

3.3.3 Image generation 24

Results and Performance 26

4.1 Client-side results and performance 26

4.1.1 Google Maps and AJAX 26

4.1.2 User Interface 26

4.2 Server-side results and performance 28

4.2.1 Database access 28

4.2.2 Data interpolation models 28

4.2.3 Image generation 38
4.2.3 Other considerations 38
Conclusions 39
5.1 Future work 39
5.2 Final words 39
Bibliography 40
Appendix 44
Appendix A: Installation Guide 44
Appendix B: File Checkout 45

Introduction

In recent years, there has been a growing awareness regarding air pollution and the need for proper monitoring and forecasting of urban air quality around the world. Air quality has worsened in many developing countries, with the escalating number of motor vehicles and movement towards wide-scale industrialization [1]. This leads to increased health risks, particularly for cardiovascular and respiratory illnesses such as asthma and lung cancer. The World Health Organisation (WHO) estimates that air pollution is the cause of two million deaths worldwide each year [2]. The air may be relatively cleaner in Australia, but air pollution still poses a big problem and costs the government of New South Wales about 4.7 billion dollars per year in medical fees [3].

As such, the Haze Watch project has been initiated to collect data regarding air pollution around Sydney using mobile wireless sensors, as well as to distribute and display the latest air quality information on various applications. This thesis, originally titled *GIS tools for Web Applications*, will focus on the latter component, aiming to build a web geographic information system (GIS) which can store, model, and represent geographically referenced air pollution data. Google Maps acts as a primary viewing platform for a graphical visualisation of Sydney's air quality.

The objectives of this thesis are:

- To build and test a fast and efficient web application that has a friendly and easy to understand user interface (UI)
- To supply accurate and reliable air pollution estimates using selected data interpolation models
- To provide dynamically constructed real-time graphical visualisations based on collected and interpolated air pollution data

The current trend of GIS technology towards online networks has made such systems even more powerful than before, as an up-to-date representation of collected data makes analysis over the entire location range much easier [4]. More people will be able to access information about air quality (which would have previously been unavailable). This ties in to one of Haze Watch's main aims of identifying pollution hotspots in Sydney and informing individuals of their personal exposure to air pollution.

Background

2.1 Haze Watch

2.1.1 Objectives of the project

The Haze Watch [5] project was recently started this year, in response to major public policy concerns regarding exposure to air pollution within urban city centres. The general goal of the project is to develop a system which collects air pollution readings for several pollutants at a high spatial resolution, and also analyse and display air pollution data for anyone with Internet access. This data may then be used to quantitatively estimate the pollution exposure of individuals. It is based on the concept of participatory sensing, by linking sensor devices to widely-available smart phones. There are some projects with similar aims, such as CamMobSens [6] and MAQUMON [7], but they are currently not ready for full release.

2.1.2 Project topology

There are two main branches of development, which can be approximately divided into a collection element and an analytic element, as seen in Figure 1. James Carrapetta [8] is handling the collection element which involves the building and manufacture of portable devices that measure the concentration of toxic gases, such as nitrogen dioxide (NO_2), sulphur dioxide (SO_2) and ozone (O_3) near ground level. These measurements are then passed on to a user's smart phone through a Bluetooth connection. At the moment of communication, the smart phone also records the sensor's position and time. Further transmissions of collected data may then be made through the mobile phone network to be stored on a remote server.

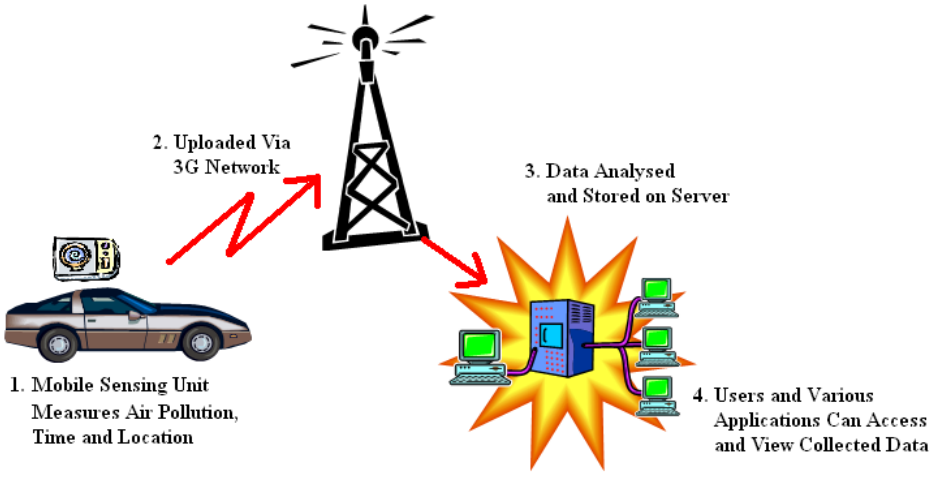


FIGURE 1: PROJECT TOPOLOGY OF HAZE WATCH

For the analytic component, a web application allows user input to access the shared database of geo-tagged data and display a specific dataset dependent on position, time and type of pollutant. Map overlays representing the range of estimated pollutant values are dynamically generated using common data interpolation models such as inverse distance weighting and Kriging over a selected gridded area of Sydney. A shared internal code interface was created to facilitate development of other applications that require the same functionality of estimating air pollution values based on nearby measured samples. Nik Youdale [9] has set up a database populated by half-hourly updates on the air quality at fifteen fixed sites around NSW from the Department of Environment and Climate Change (DECC) website [10], and has also developed an iPhone application to predict an individual's exposure to air pollution.

It is crucial that both components work in tandem with each other, as a vast collection of data will require some form of visualisation to assist in understanding, while the analytic component needs a large and well distributed dataset to ensure reliability and consistency of our estimated air pollution values.

2.2 Similar products

In addition to related projects focused on air pollution like CamMobSens, this thesis also shares similarities with other web-based geographic information systems (GIS) such as Lifemapper.org [11], which disseminates graphical visualisations of an extensive collection of biological data. A broadly accepted definition of a GIS [12] describes it as:

> A system of hardware, software and procedures to facilitate the management, manipulation, analysis, modelling, representation and display of geo-referenced data to solve complex problems regarding planning and management of resources.

To put it simply, a GIS application deals with the convergence of cartography and a database of position-based values. Its value lies in easily summarising and communicating data to people through the use of visual images and maps. GIS software such as Gamma Designs' GS+ 9 [13] and Golden Software's Surfer 9 [14] are highly sought after and cost in the hundreds, though there are also open source alternatives such as Geographic Resources Analysis Support System (GRASS) GIS [15] and Quantum GIS [16], which are founding projects of the Open Source Geospatial Foundation (OSGeo).

Peng and Tsou [17] further elaborate on systems with respect to a web GIS, also known as an Internet GIS. They describe it as a GIS centred on the use of Internet technology and relies on real-time data analysis carried out on open distributed networks. Users will be able to directly access the application through their web browser, and be supplied with the latest information. Consequently, web GIS is seen as the next progressive step up from traditional desktop-based proprietary GIS programs.

There are a few GIS web tools available on the Internet, which include the commercially available ArcGIS Server [18] and the open source MapServer and GeoServer [19]. Google Maps is known as a web mapping service, not as a web GIS per se, but because we plan to integrate it with the storage and analysis of Haze Watch's geo-referenced air pollution data, the entire system can be considered as one. Hence we have strived to emulate the accuracy of full GIS application suites, and also to offer online data sharing and distribution capabilities.

2.3 Data interpolation techniques

One of the most important components of this web application is the air quality models needed to predict the different concentrations of air pollution at a certain space and time. This will determine the reliability and accuracy of any derived data and therefore the quality of our project. Based on the principle that the data value measured at a location will be similar to other nearby positions and that this relationship can be modelled, we use two-dimensional interpolation to determine the concentration of air pollution at points not within the dataset, thereby extending usability beyond the sampled sites.

Even so, there are many different forms of interpolation which process data differently. The speed and accuracy of various techniques, as well as the variability and density of the original dataset [20], must all be taken into account. In particular, we note that interpolated data has a greater reliability given that sampled data locations are densely and uniformly distributed, but conversely, if data locations are clustered with large gaps between sites, inaccurate estimates will be obtained [21]. This holds true regardless of the method we choose. We must also be aware of the fact that interpolation inherently underestimates the peaks and overestimates the dips due to the nature of averaging.

Inverse distance weighting (IDW) and kriging are two of the most common mathematical interpolation techniques, and are examples of the choices to be made between speed, complexity and accuracy. As they have been chosen for implementation in this thesis, the theory behind the algorithms is discussed in the following sections.

2.3.1 Inverse distance weighting interpolation

Inverse distance weighting involves the allocation of weights based on the distances between positions that have known values and the positions to be predicted. Shepard's formula is as stated [22]:

$$f(x) = \frac{\sum_i w_i(x) y_i}{\sum_i w_i(x)}, \quad w_i(x) = \left(\frac{1}{\|x - x_i\|} \right)^p, \quad p = 2$$

As a point gets further away from the interpolated position, it becomes less significant in the calculation, and hence its weight in the total equation is reduced. The power, P, determines the

degree of diminishing significance, with higher powers giving greater emphasis on close neighbouring points.

IDW can be implemented quite easily, and the final predicted value is also computed quickly. However, it also has a high error rate, particularly when points are sparsely distributed because a far-away point may have too high an impact. The contour maps generated from a grid of IDW values are not very smooth, with a very sharp gradient seen near an actual measuring site (also known as a bull's-eye effect). Any point that is outside the range of measured sites is subject to inaccuracy, due to IDW's inability to extrapolate information outside of the given data set.

2.3.2 Kriging interpolation

Kriging is a more complex method of interpolation, but also promises more robust results. It is similar to the IDW method, as weights are assigned according to surrounding measured values. The difference is that weights are defined using the statistical variance between two points, which is a measure of their spatial autocorrelation. Semi-variance is calculated using the formula shown below [23]:

$$\gamma^*(h) = \frac{1}{2n}\sum[g(x) - g(x+h)]^2$$

What it means is that for each pair of locations separated by a distance h, we calculate the difference squared between their values. Since there are typically many pairs with unique distances, they are grouped into lag bins (eg. all pairs of points that are more than 20m apart but less than 30m are clustered together). We average the semi-variance values for all pairs within a lag bin, do the same for the other distances, and then graph them to form an empirical semivariogram. Thus a semivariogram relies on a stochastic interpretation of the data set, as compared to the previous deterministic IDW model [24].

We still need to fit a model to the empirical semivariogram in order to cover all possible distances (as shown in Figure 2), especially if the data for lags close to zero is sparse.

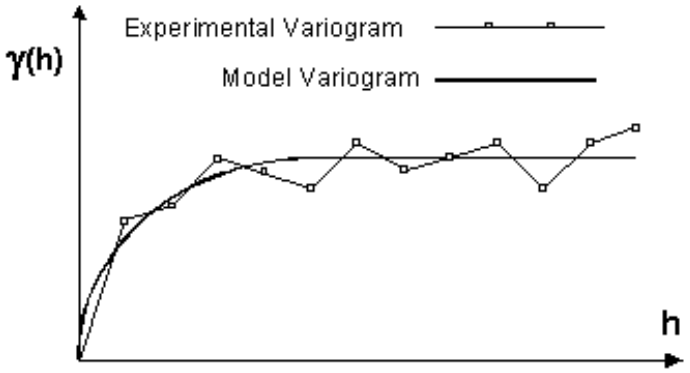


FIGURE 2: EXPERIMENTAL AND MODEL VARIOGRAM FOR KRIGING

The choice of model (Figure 3) relies on the underlying experimental variogram after considering their nugget value, range and sill. The nugget effect is equal to the y-axis intercept of the model and should technically be zero, but signifies measurement errors or unaccounted variances when it is non-zero. The range is the distance where the graph evens out, while the sill is the y-value where the range is reached. A linear model should be used if the experimental variogram does not flatten out. Both the exponential and spherical models level out after a certain distance, implying that locations further apart than that distance are no longer related. Otherwise, their main differences lie in how sharply the graph changes.

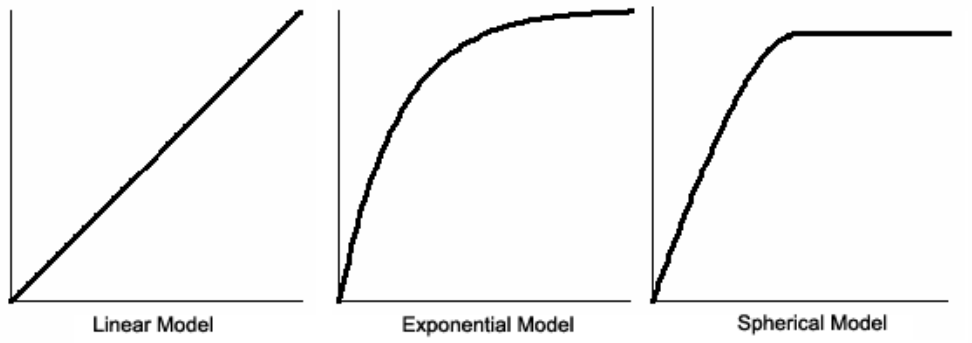


FIGURE 3: MODEL VARIOGRAMS FOR KRIGING [25]

The values can then be predicted using weights in relation to this fitted variogram, using simple, ordinary or universal kriging. The different types of kriging place different constraints on the weights. Simple kriging assumes that there is a known constant trend in the data, whereas universal kriging supposes a general linear trend model. Ordinary kriging assumes the constant mean is unknown, which is a reasonable mathematical conjecture and so will be used in our implementation. All mentions about kriging hereafter refer to ordinary kriging. The equation

below shows the equation for the kriging estimated value, with w_i representing the weight given to each sampled point, $Z(x_i)$.

$$\hat{Z}(x_0) = \sum_{i=1}^{n} w_i(x_0) Z(x_i) \quad \text{where} \quad \sum_{i=1}^{n} w_i(x_0) = \mathbf{1}$$

To find the weights, w_1 to w_n, we must solve the following system of linear equations [26] using matrix inversion, where $\gamma(d_{ij})$ is the semi-variance for d_{ij} determined from the model variogram, and λ is the Lagrange multiplier used to minimise error, such that $\lambda(x) = \lambda$ (ie. the unknown trend).

$$\begin{bmatrix} \gamma(d_{11}) & \gamma(d_{12}) & \cdots & \gamma(d_{1n}) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \gamma(d_{n1}) & \gamma(d_{n2}) & \cdots & \gamma(d_{nn}) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix} = \begin{bmatrix} w_1 \\ \vdots \\ w_n \\ \lambda \end{bmatrix} \begin{bmatrix} \gamma(d_{10}) \\ \vdots \\ \gamma(d_{n0}) \\ 1 \end{bmatrix}$$

For the final interpolated value and variance [26]:

$$\hat{Z}(x_0) = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}' \begin{pmatrix} Z(x_1) \\ \vdots \\ Z(x_n) \end{pmatrix}$$

$$var\left(\hat{Z}(x_0) - Z(x_0)\right) = \begin{pmatrix} w_1 \\ \vdots \\ w_n \\ \lambda \end{pmatrix}' \begin{pmatrix} \gamma(d_{10}) \\ \vdots \\ \gamma(d_{n0}) \\ 1 \end{pmatrix}$$

The great benefit of a stochastic technique is that it provides the ability to assess error or uncertainty of the estimated point. Therefore, the kriging interpolation is a widely accepted method to model the quality of air in many studies. It also presents a much smoother and natural-looking contour plot generated from interpolated data.

Implementation

3.1 System overview

The web application could be divided into two main sections, consisting of a client-side component and a server-side component which are separated by a network, as shown in Figure 4. The server stores geo-referenced data, keeping track of measured pollutant values at different positions and times. It also interpolates data and generates a contour map for selected datasets. The client side allows users to enter position and time parameters and pass those to the server. Another task is to display the Google Maps tile layers and also the generated contour map as an overlay.

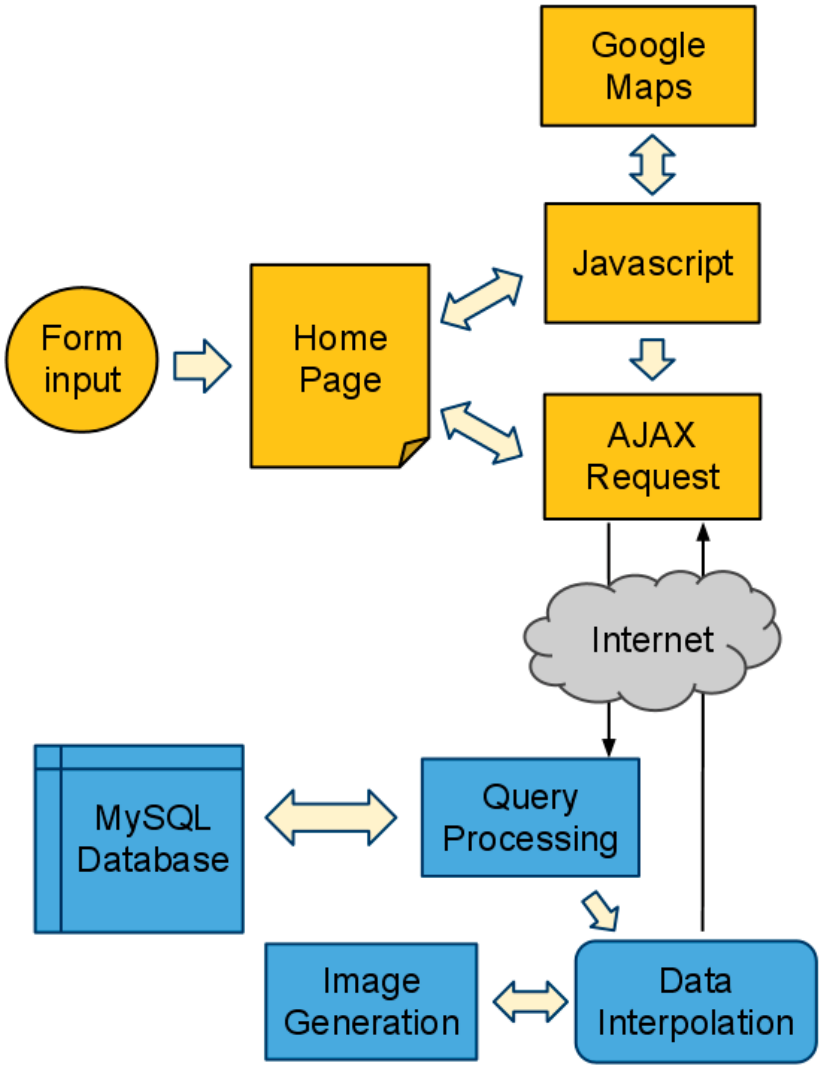


FIGURE 4: DIAGRAM ILLUSTRATING SYSTEM COMPONENTS AND FLOW

Furthermore, the server side has a standardised code interface (*model.php*) to be shared with the iPhone application that Youdale [9] is making, as shown below.

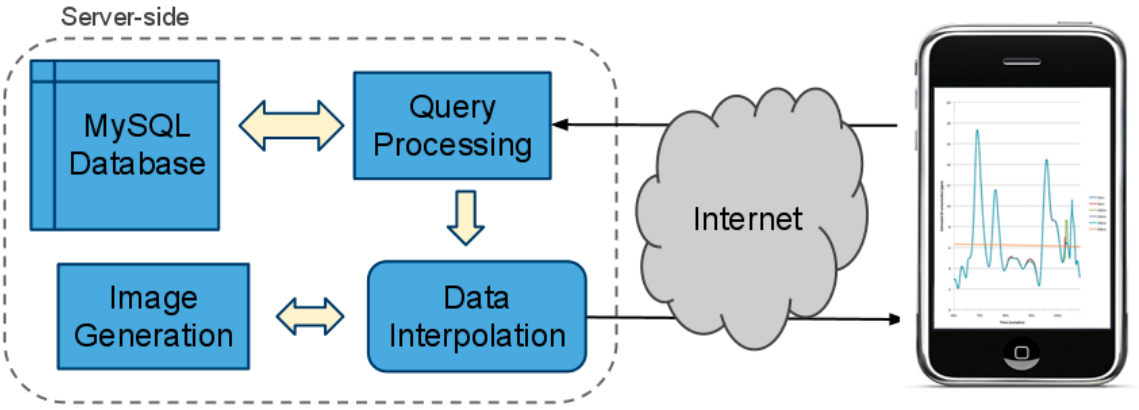


FIGURE 5: SERVER-SIDE INTERACTION WITH IPHONE APPLICATION

3.2 Client-side implementation

As a web application, the frontend will be widely available to everyone, as long as they have an Internet connection and compliant browser. We want it to be fast, intuitive and user-friendly, but also to leverage on the power of server-side applications to deliver the maximum data modelling and visualisation capabilities. The standard web technologies of HTML, CSS and Javascript are all used to develop the web application, as well as the burgeoning growth of AJAX (asynchronous JavaScript and XML) in hand with PHP server-side scripting. Separate CSS files allows for trouble-free change in the appearance of the web tool.

3.2.1 Google Maps and AJAX

Google Maps is a well-known web tool launched in February 2005, which marked a breakthrough for web cartography when their tile-based implementation of maps proved to be much quicker and intuitive for users. The company then made their application programming interface (API) freely available, allowing other web developers to use Google Maps to share individualized content and combine different datasets on a single map. This ease of use, internet popularity and clearly documented API were factors in the decision to use Google Maps for the web application. Another close choice would be Google Earth, which has more advanced features, however less people use it and an additional plug-in would be required.

The third and newest form of the Google Maps Javascript API [27] was released at the start of 2010. Version 3 maps promise faster load times, especially so for smartphones due to the removal of bloated code. Google announced the deprecation of Version 2 maps in May 2010, and moved support over to V3, the current official version of the Google Maps Javascript API. This indicates that expected lifespan of V3 maps is about three years, and acts as an example of how quickly web technologies change, and demonstrates that updates to our web application must be made on a regular long-term basis.

AJAX technologies have grown in popularity in recent years, as web developers make full use of the interactivity it provides. With the advent of AJAX, a single client action no longer has to reload the entire page in order to display any new changes, dramatically improving the user experience [28]. The use of JavaScript and the XMLHttpRequest object allows data to be exchanged asynchronously between the web browser and server, while the Document Object

Model (DOM) is utilised to dynamically handle data. An AJAX interface is harder to implement compared to a static page, but the greater functionality is worth the time and effort.

Google Maps in particular relies heavily on such AJAX techniques, and our web application seamlessly integrates these into our system, from the handling of form inputs, dynamic estimation of pollutant values to loading images. The following figure illustrates some of these basic Javascript components.

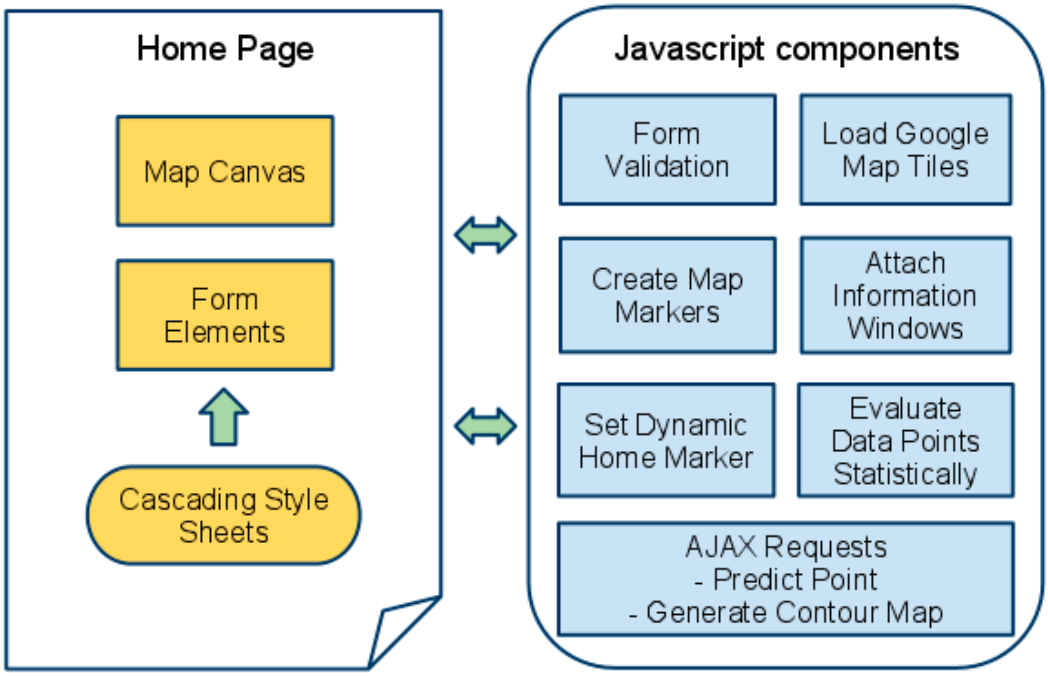


FIGURE 6: DIAGRAM OF JAVASCRIPT COMPONENTS

3.2.2 User interface

At the end of Session 1 2010, the interface was still fairly basic as seen in Figure 7, with more screen estate given to the map and a small footer containing form elements. The map is sized proportionately to the dimensions of the user's browser window, thereby extending across the entire screen. This placed greater emphasis on the graphical representation of selected data.

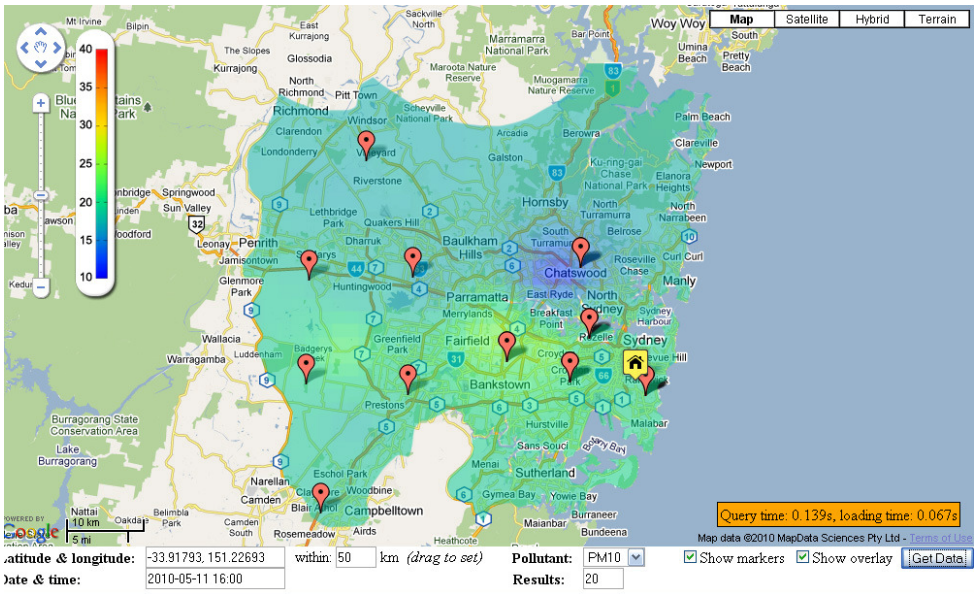


FIGURE 7: PAST WEB APPLICATION USER INTERFACE

However, the decision to add increased functionality through the form inputs meant that it was more appropriate to move the form elements to form a separate sidebar (see Figure 8). This proves to be more intuitive to a user's eye, as we are naturally conditioned to fill in forms vertically rather than horizontally. The layout of the form also becomes much cleaner and easier to navigate.

We allow the user to tailor their query by inputting values for latitude, longitude, search radius, date and time and pollutant type. A draggable home marker designates the user's point of interest, such that one does not have to manually enter in explicit latitudes and longitudes (which are normally not well known). jQuery UI [29], a well-known Javascript API plug-in, is used as it simplifies client-side scripting and provides high-level widgets such as a date-picker and slider bar, both of which provide advanced interactivity for the web application. The date-picker is important because users can select their wanted date through a calendar, especially since entering it manually is slightly confusing due to the database formatted date. The same applies for the slider bar, which lets users adjust the time by sliding the knob rather than typing in the time in 24-hour format by hand.

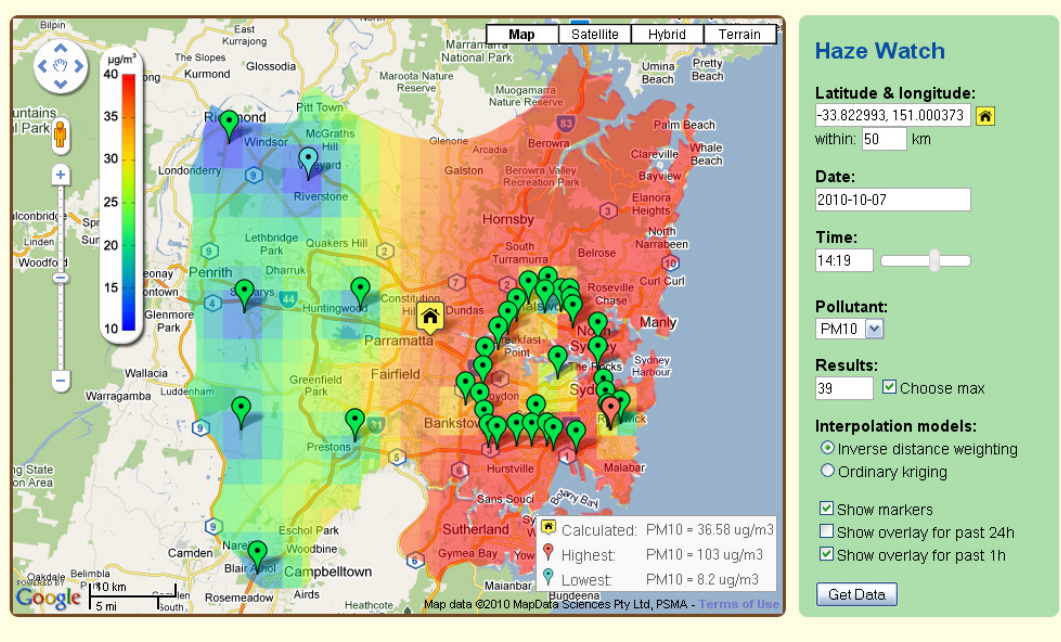


FIGURE 8: CURRENT WEB APPLICATION USER INTERFACE

The types of pollutants are listed in a drop-down menu as it offers a cleaner user interface. The number of neighbouring results is a significant value as it can greatly affect the final estimated results. Hence we give people the option of changing the number by themselves, but still set the default to finding the maximum possible set of sampled sites which meet their initial query standards. Two different methods of interpolation (IDW and kriging) are also presented to the user to choose.

Currently there are two main display variables, one which will only show points where data was collected, while the other is an option to display a gradient-filled contour map of predicted pollutant values. Mousing over the marker associated with the sampled site will bring up the measured pollutant data in an information window, which reduces on-screen clutter yet gives a convenient way of looking at point data. Both display options offer colour-coded indicators which immediately inform users of unsafe levels of pollution. If we display sampled sites as points, a light-blue marker indicates the site with the smallest sampled value, while a red marker denotes the site with the highest. For the contour map, there is an associated colour bar with blue signifying lower pollutant values and red indicating higher pollutant values.

3.3 Server-side implementation

3.3.1 Database access

PHP is a scripting language built from the ground up for web development and has specialised functions for interaction with the MySQL database. These attributes make it a superior candidate for our server-side based programming over others such as Perl. Form inputs are sent using the get method to a specific PHP file for query processing. The PHP get method is useful for database querying, especially for testing purposes as we are able to bookmark pages or alter the query directly in the URL bar. An example URL is "form-contourmap.php?lat=-33.91793&lng=151.22693&radius=50&datetime=201010081200&results=20&pollutant=pm10&maxfit=true&modeltype=idw".

The PHP file parses the terms into a structure suitable for MySQL. It then opens a connection to the database and retrieves the relevant dataset. These could be sent back to the client-side for the display of individual markers of measuring sites, or to a data interpolation model to undergo more processing.

The specific database query which selects the neighbouring data points also particularly influences the final estimated value, almost as much as the interpolation method itself. In this case, we pick points on the basis of a nearest neighbour search, such that they are the closest possible points in terms of distance and time. Preference is given to closest time and then distance. We do disregard the third dimension characteristic of topographic height, so distance is assessed purely on their latitude and longitude parameters. Temporal aspects are mostly ignored other than setting an initial window of two hours before and after the requested time, because pollutants are highly variant according to dispersion properties and weather factors. However, these two assumptions could be modified after further research.

Some initial filtering of the retrieved dataset is carried out during the database query, chiefly for latitude and longitude points. The interpolation algorithms do not work properly when the exact same points hold different pollutant values. This occurs either when the site is sampled at different times or the spatial difference between two points is negligible, making them appear to be only a single point. Hence we filter the results by rounding the latitude and longitude parameters to two decimal places (corresponding to a precision of roughly 1 kilometre, ie. only differentiating between points that are more than 1km apart).

3.3.2 Data interpolation models

Inverse distance weighting

The algorithm is written using PHP, as it is straightforward enough to implement, and fast enough that the time difference due to using an interpreted language is negligible. The value to be estimated is consistently iterated over the number of points within the array of sampled sites retrieved from the database. Attempting to stop when the percentage change of the predicted value is deemed insignificant either has no chance to occur due to the variability of data, or creates a large unexpected deviance in the final estimate, thus it is recommended that the final estimated value is only outputted when all measured sites have been weighted accordingly.

Kriging interpolation

Firstly, semivariograms were constructed for two different datasets taken with the same parameters other than number of neighbouring points and type of pollutant (see Figure 9). 35 measured points were obtained from the database for the pollutant PM10, while 68 points were taken for CO. The numbers above each point on the graph indicate the number of distance pairs inside the corresponding lag bin. The discrepancies between these two variograms suggest that it would be necessary to fit the model variogram for each different pollutant. There is no clear sill or range for either of these variograms, but nugget effect appears to be zero.

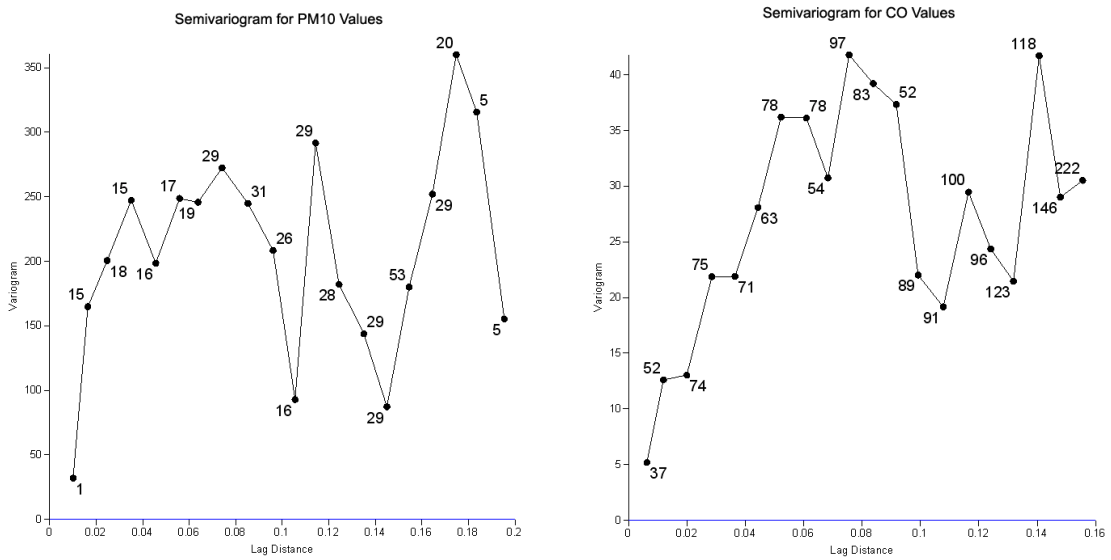


FIGURE 9: EMPIRICAL SEMIVARIOGRAM FOR PM10 (LEFT) AND CO (RIGHT)

We use a CGI script (*krigingR.cgi*) to pipe instructions to *R* [30] which is a long-standing statistical computing language and system, and is considered to be the major standard amongst

statisticians, engineers and scientists for analysing data. Given *R*'s longevity, many packages and plug-ins are available for use, chief amongst them *gstat* [31] and *automap* [32]. *gstat* gives us the ability to fit models to variograms and graph the result, as well as to predict the kriging estimate, while *automap* performs automatic calculations based on the dataset in order to find the best-fitting variograms.

A linear model was initially fitted for the PM10 variogram (Figure 10 left), before trying the automatic fit function in *automap (*Figure 10 left). Note that some lag bins are discarded to get a better fit. The fitted variogram model for CO is also shown below. After finalising our variogram, we can then use *gstat* to predict our desired kriging estimator. The estimated value together with its variance is written to an external text file, which is read by a PHP script to be displayed in the web application, thereby achieving an ordinary kriging data interpolation model.

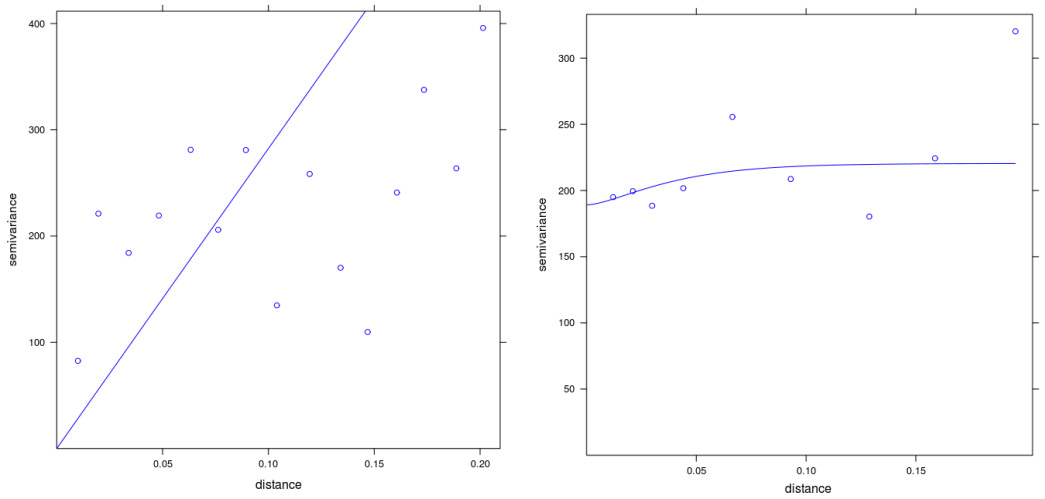


FIGURE 10: LINEAR VARIOGRAM (LEFT) AND EXPONENTIAL VARIOGRAM (RIGHT) FOR PM10

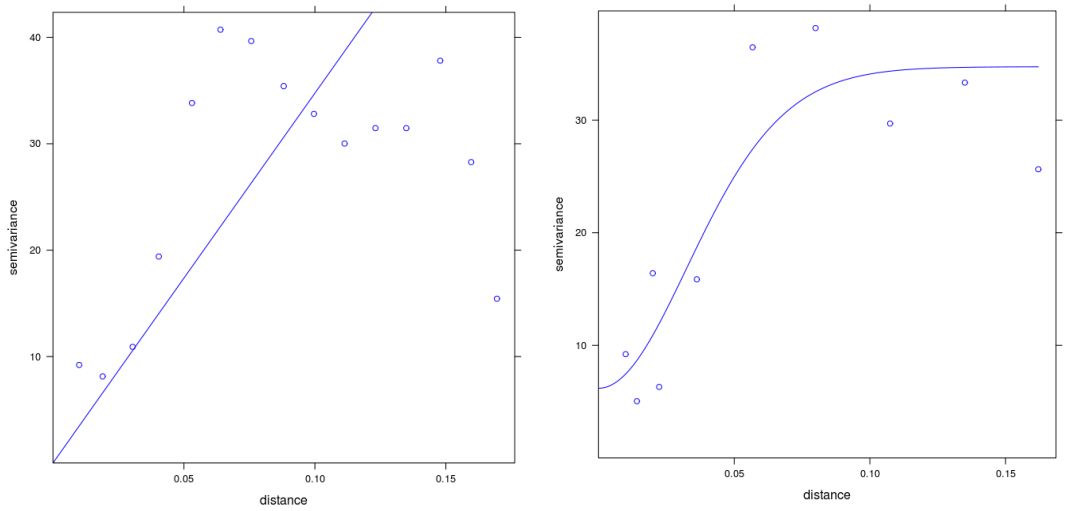


FIGURE 11: LINEAR VARIOGRAM (LEFT) AND EXPONENTIAL VARIOGRAM (RIGHT) FOR CO

3.3.3 Image generation

Image generation poses several distinctive challenges. First of all, note that we will need to apply a mask (Figure 13 left) to any generated image so as to only show the desired area. Secondly, we have to make a choice between using raster datasets or vector images [33], which will each have unique ways of creation and display.

With vector images, we need an algorithm which can determine the spatial relationships between each cell, and then generate lines around similarly-classified cells. These lines would then be known as isopleths and could possibly form a closed polygon shape. By colouring in those shapes, we have a clearly defined contour map. For raster datasets, data interpolation is carried out on a whole rectangular grid, producing a predicted pollutant value for each individual cell. We then fill in the grid with different colours depending on those predicted values, akin to painting pixels.

Vector imaging was deemed too complex at present, with raster images easier to be set up and produced. A Common Gateway Interface (CGI) script, *contour.cgi*, was employed to call GNUplot [34] as an external program. GNUplot is an open-source function plotting program and is used here as an image generation backend by utilising the three-dimensional plotting tool (Figure 12 left), and flattening the resultant plot (Figure 12 right) thereby creating a satisfactory gradient-filled image.

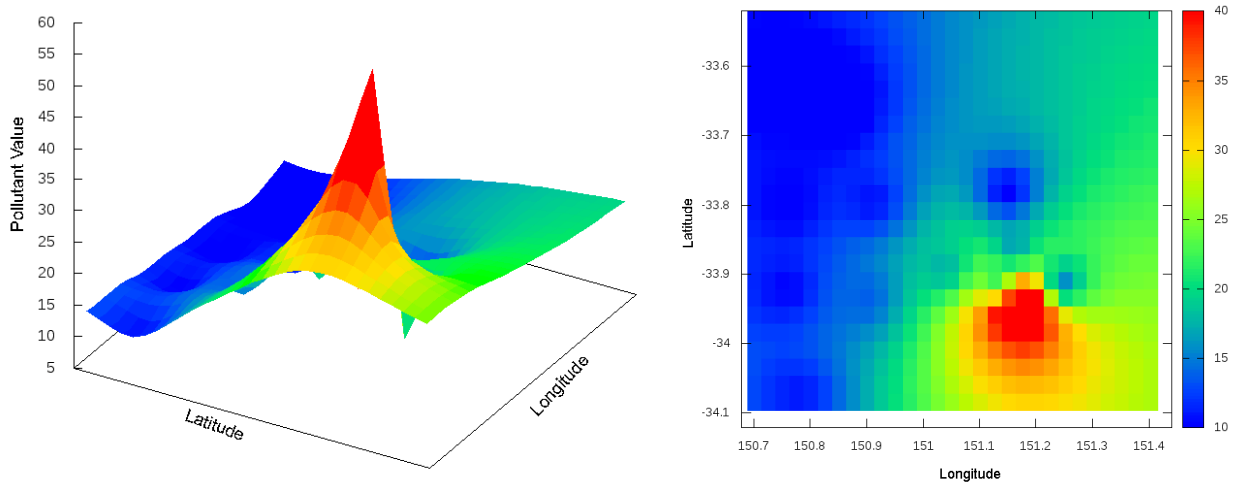


FIGURE 12: 3D PLOT OF PM10 POLLUTANT VALUES (LEFT); IMAGE GENERATED FROM GNUPLOT (RIGHT)

The range and colour palette of the plot, size of the desired image can all be customised. We set the range to be the latitudes and longitudes of the outskirts of greater Sydney, while the colour palette is adjusted to signify hazardous levels of pollutant exposure as red (typical danger colour), and safer level to be blue. This is changed accordingly to the type of pollutant being plotted, since unsafe levels differ from pollutant to pollutant. Current World Health Organisation (WHO) guidelines [35] to safe pollutant thresholds are shown in Table 1.

TABLE 1: SAFE POLLUTANT LEVELS

Pollutant	**Guideline values**
PM10	50 µg/m^3 for a 24-hour mean
PM2.5	25 µg/m^3 for a 24-hour mean
Ozone (O_3)	0.05 ppm for a 24-hour mean
Nitrogen dioxide (NO_2)	0.25 ppm for a 1-hour mean
Sulphur dioxide (SO_2)	5 ppm for a 1-hour mean
Carbon Monoxide (CO)	35 ppm for a 8-hour mean

The GD library [36] of PHP was then used to combine both the mask and GNUplot-created image into a single image as shown in Figure 13. The main problem with GNUplot is that it takes in datasets through text files, necessitating the creation of a specially formatted temporary text file for the latitudes, longitudes and pollutant values to be graphed.

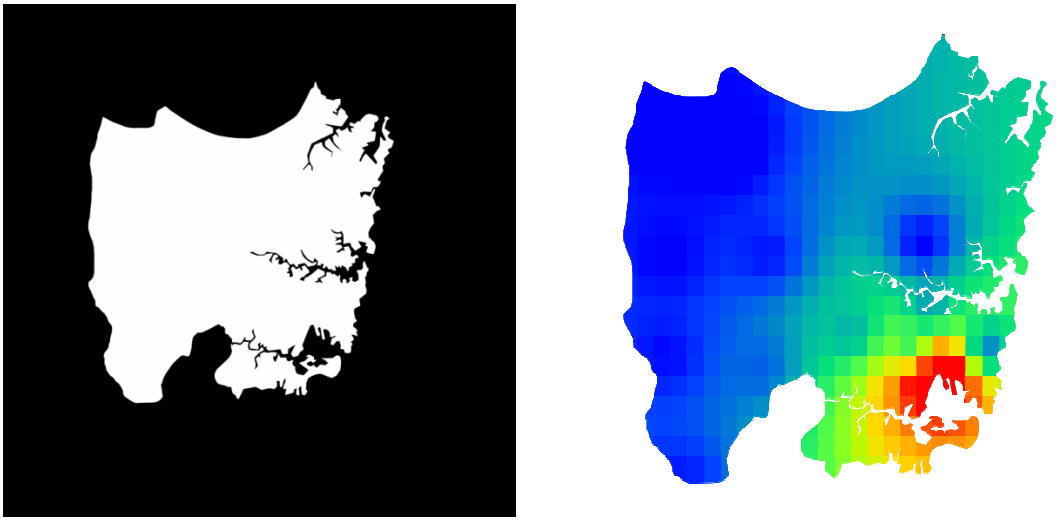


FIGURE 13: MASK OF GREATER SYDNEY (LEFT); MASKED CONTOUR MAP (RIGHT)

Results and Performance

4.1 Client-side results and performance

4.1.1 Google Maps and AJAX

The performance of Javascript and thus the overall loading time of the web application differ from browser to browser. The worst loading times were experienced in Internet Explorer, with Google Chrome beating out Firefox in terms of speed. In order to further reduce loading times, algorithms should be made as efficient as possible with minimal computation occurring, while carrying out most processing on the server. The main Javascript functions were maintained in a single file, *loadmarkers_v2.js*, thereby reducing the amount of overhead for the browser to load, as compared to opening connections for several small scripts.

Since the contour map is requested through AJAX, and does not require a refresh of the page to load, users are free to navigate through the map features while waiting for the image. This is essential since there is a feeling of action rather than inaction and keeps the user engaged in the web application.

4.1.2 User Interface

One of the key challenges of the user interface is to ensure that it is cross-compatible across different browsers such that the fundamental look of the website is maintained. The main development work was carried out in Mozilla Firefox 3.6, but other versions and brands of web browsers were also tested. There is a unified coherent look for Firefox and Google Chrome, given that they follow web compatibility standards. Unfortunately, Internet Explorer does not adhere completely, resulting in little quirks of design such as non-support for rounded corners and a different transparency setting. Some workarounds (like specially adjusting the opacity values of the generated contour map) had to be specifically implemented in order to make the web application presentable. This is essential because Internet Explorer still holds roughly 50% of browser usage share, compared to 30% for Firefox and 11.5% for Chrome [37].

The web application, although much improved from Session 1 2010, is still fairly unintuitive, especially if the user is unfamiliar with geostatistics. Since the Haze Watch project is primarily targeted towards consumers, more work must be done to ease the use of the interface, perhaps

with increased instructions and advice on how the different parameters affect the final estimated results. We must balance the need to simplify the web tool and yet provide enough complexity such that advanced users can fully utilise it.

4.2 Server-side results and performance

4.2.1 Database access

The phrasing of mySQL queries is crucial in ensuring fast retrieval times, with simpler requests returning more quickly. However, it is also much easier to carry out some pre-filtering at the database rather than filter the database results as-is due to specialised database functions that enables quicker grouping and sorting. This leads to faster processing later on, especially with regards to data interpolation. The same query results are shared between different functions, so there is little need to keep opening database connections and in doing so, reduces the network traffic that could severely slow down the database.

The database usually caches the most recent or most popular searches, which speeds up the query retrieval time if two users happen to call for the same dataset. This should not happen that often, but more thought could be put into how to harness this to our advantage, for example, making more general and less specific queries even as it requires us to lower the sensitivity of the search. As the database expands, general searches will probably be more time-efficient even with a trade-off in later data processing.

4.2.2 Data interpolation models

When evaluating the reliability and accuracy of data interpolation algorithms, it is extremely important to remember the salient nature of errors in geo-referenced data. Besides modelling errors, there will also be errors in measurement and errors in positioning, which could result in a larger than expected data variations [38]. The models are only as good as the data they are built upon. In the following sections, we assume that there are no measurement or positioning errors and focus on modelling errors.

The cross-validation method, otherwise known as the leave-one-out method, is generally used to calculate the accuracy of interpolation. The method is based on taking away one data point at a time and then estimating the value at the location of the removed point using the left over samples. This acts as though the removed point does not exist. We then calculate the residual between the actual measured value and the new value interpolated from the remaining samples. This is then repeated until every sample has been removed. The root-mean of the squared residuals (RMSE) signify the overall performance of the data interpolation model. The equation below is an example of how to calculate RMSE [39].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (Z_{i(\text{int})} - Z_i)^2}{n}}$$

RMSE is an absolute measure of fit, indicating the fit of the model to the data (ie. how close the sampled data points are to the model's predicted values). It can be understood as the standard deviation of the unexplained variance between the actual measurements and estimated measurements. Low values of RMSE are generally better because it signifies little difference in the residuals. We also need to consider the variability of the underlying data, wherein if the standard deviation is greater than the RMSE, then we are overestimating the variability of our predictions, but if standard deviation is less than the RMSE, the variability of the predictions is underestimated [40].

Different accuracy versus number of points on the map

Calculations of standard deviation and RMSE were carried out on the dataset of PM10 retrieved on the 7th of October 2010 at 2pm, and are tabulated below. The results indicate that RMSE values are innately linked to the standard deviation of the observed values, as shown in Figure 14 and 15. Since the NRMSE is closest to unity when interpolation is carried out on a sample dataset of 30, it indicates that this is one of the most reliable results.

TABLE 2: TABULATION OF RMSE OF IDW MODEL

No. of measured points	Standard deviation (ug/m3)	RMSE(ug/m3)	Normalised RMSE
5	2.91108	3.35341	1.15195
10	3.04867	2.72681	0.89443
15	15.5251	11.9344	0.76872
20	17.6308	14.5721	0.82652
25	16.6731	14.3605	0.86130
30	16.3350	14.6556	0.89719
35	17.5294	15.5876	0.88922

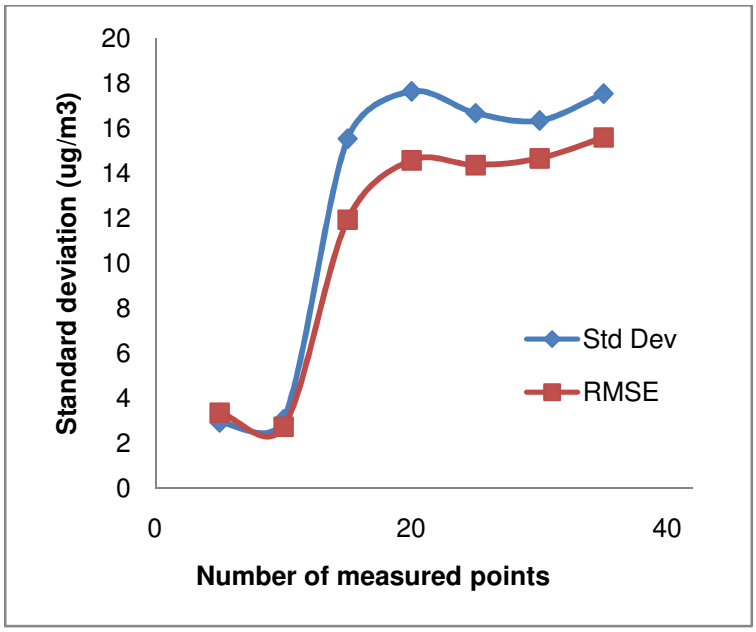


FIGURE 14: STANDARD DEVIATION AND RMSE OF DATA POINTS

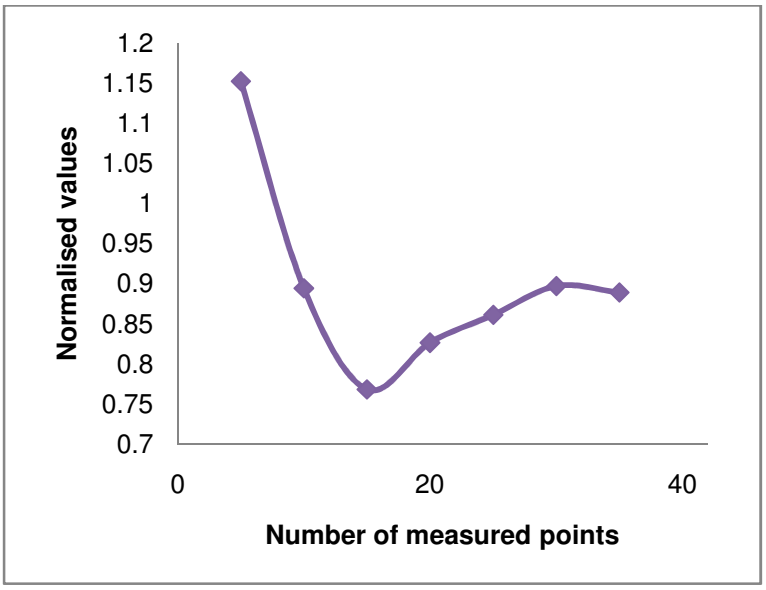


FIGURE 15: NORMALISED RMSE

Figure 16 illustrates the difference in visualisations when the number of points is varied. For a dataset of 10, the sampled sites are too sparse and have measured low values of PM10, resulting in a uniform blue. For a dataset of 35, areas outside of the range have been overly extrapolated, turning the eastern side of Sydney red and causing additional errors reflected in the RMSE.

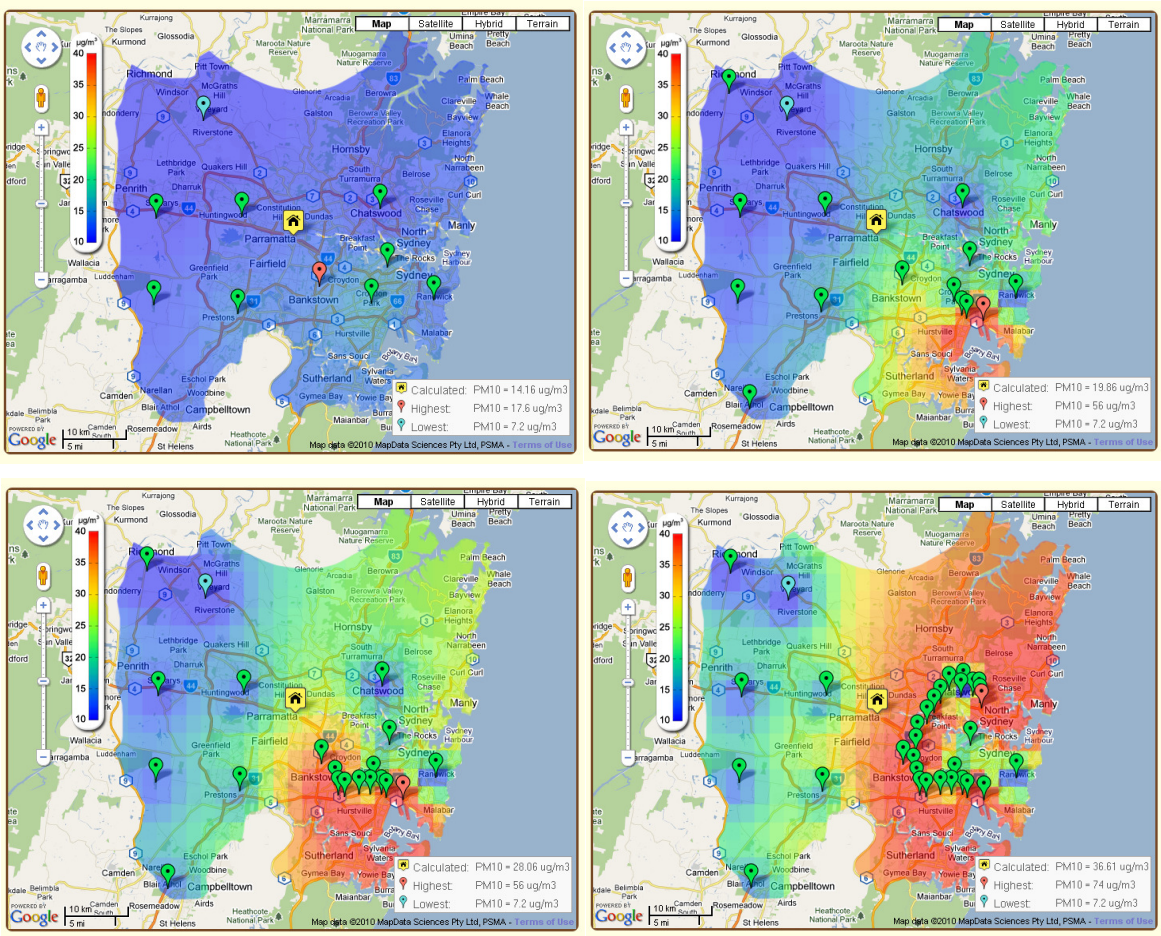


FIGURE 16: CLOCKWISE FROM UPPER LEFT, CONTOUR MAPS FOR 10 POINTS, 15 POINTS, 25 POINTS AND 35 POINTS

Accuracy difference between different algorithms

In this case, the power of the IDW interpolator was investigated by varying it from 1 to 4, and carrying out cross-validation to obtain their RMSE values.

TABLE 3: TABULATION OF STANDARD DEVIATION AND RMSE OF IDW WITH DIFFERENT POWERS

Power	Standard Deviation (ug/m3)	RMSE (ug/m3)	Normalised RMSE (ug/m3)
1	20.24313	18.97328	0.93727
2	20.24313	19.36153	0.95645
3	20.24313	19.82997	0.97959
4	20.24313	19.41376	0.95903

The default power is adjusted to two, but the power of one appears to be better due to its smaller RMSE value (albeit quite insignificantly). Figure 18 demonstrates the visual disparity between choosing different powers. For the power of 1, there is too much emphasis on points that are far away, while the opposite is true for the power of 4. Using an inverse distance squared model still seems to achieve the best balance between RMSE value and visual impact.

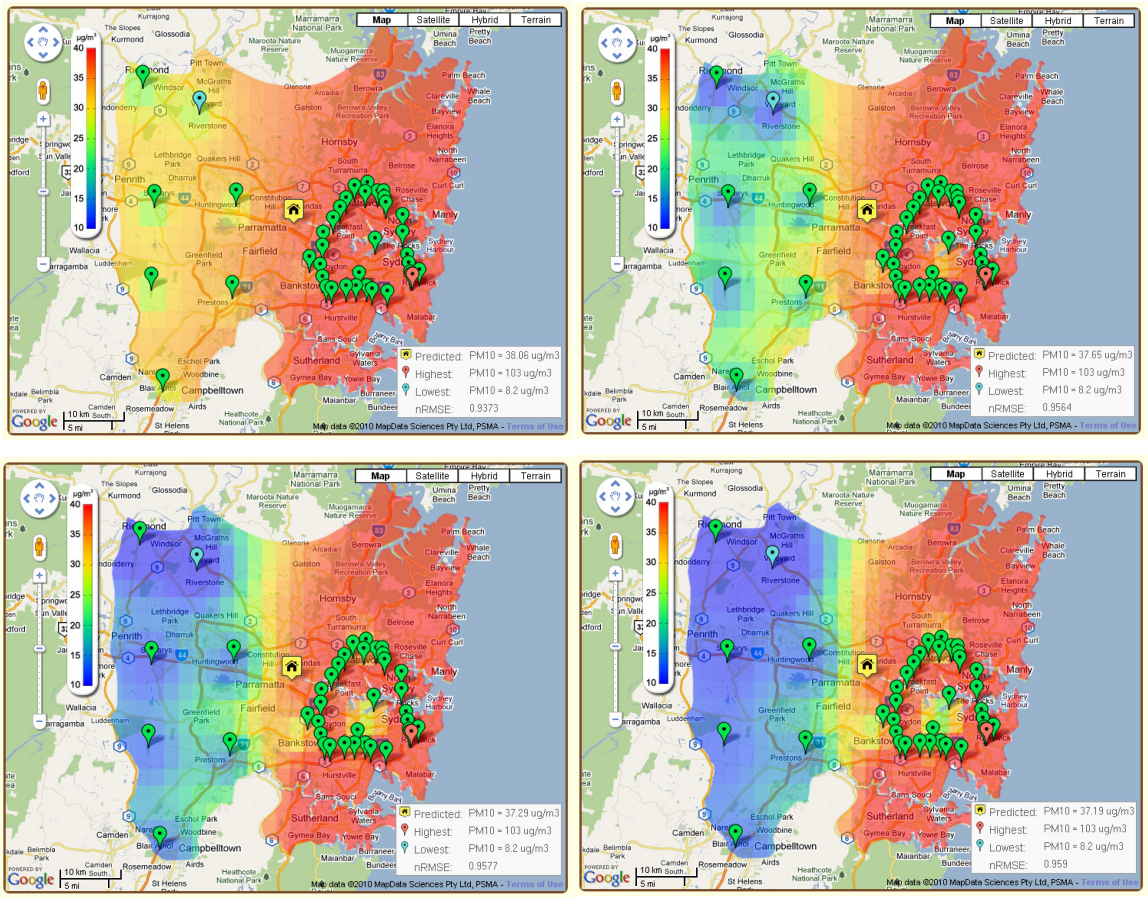


FIGURE 17: CLOCKWISE FROM UPPER LEFT, CONTOUR MAPS FOR IDW POWER 1, 2, 3 AND 4

Different time taken versus size of grid for IDW

By varying the size of the grid, we are changing the total number of points that need to be interpolated. For a 10x10 grid, one hundred points are interpolated, contrasted with 2500 for a 50x50 grid. Figure 19 shows that it is not an entirely linear relationship. Figure 20 demonstrates the stark change in grid size causes the gridded interpolation to give less quality information, such that having a 40x40 grid is a good compromise between time and sensitivity.

TABLE 4: TIME TAKEN FOR DIFFERENT GRID SIZES

Size of grid	Time Taken for Data Interpolation (ms)
10	33.142
20	104.056
30	231.701
40	394.948
50	607.203

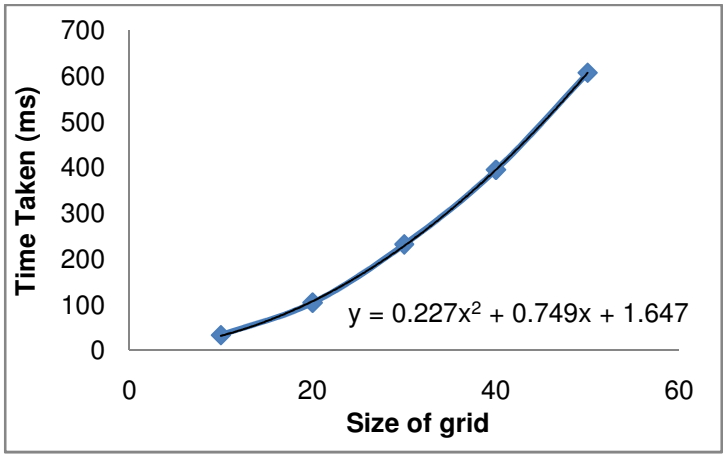


FIGURE 18: TIME TAKEN FOR DATA INTERPOLATION AGAINST SIZE OF GRID

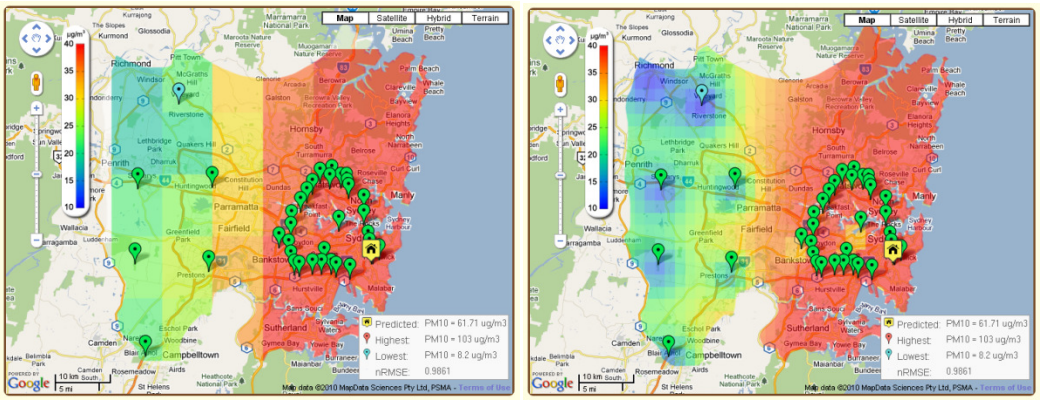


FIGURE 19: CONTOUR MAPS FOR GRID SIZE 10 (LEFT) AND 40 (RIGHT)

Different time taken versus number of points for IDW

These timings were carried out on the dataset of CO pollutant values on the 7th of October 2010 at 2 pm. The time to carry out different components of the IDW model is recorded and tabulated in Table 5 and graphed in Figure 18. Database query and image generation proves to be independent of the number of measured sites, while data interpolation scales linearly. Generating the contour map seems to be the biggest bottleneck of the system, particularly when the number of samples is so small. However, time taken for data interpolation could quickly grow into a problem, especially after 300 observed values when it is estimated to take as long as the image generation itself.

TABLE 5: BREAKDOWN OF TIMINGS FOR DIFFERENT COMPONENTS IN IDW MODEL

No. of measured points	Time Taken (ms)			
	Database query	Data interpolation	Image generation	Total
10	274.54	143.45	2943.47	3361.46
20	275.69	254.94	2938.43	3469.06
30	274.65	365.89	2951.58	3592.12
40	278.84	477.40	2958.25	3714.49
50	274.58	587.43	2984.42	3846.43
60	275.86	709.28	2933.85	3918.99
68	274.54	790.10	2943.08	4007.73

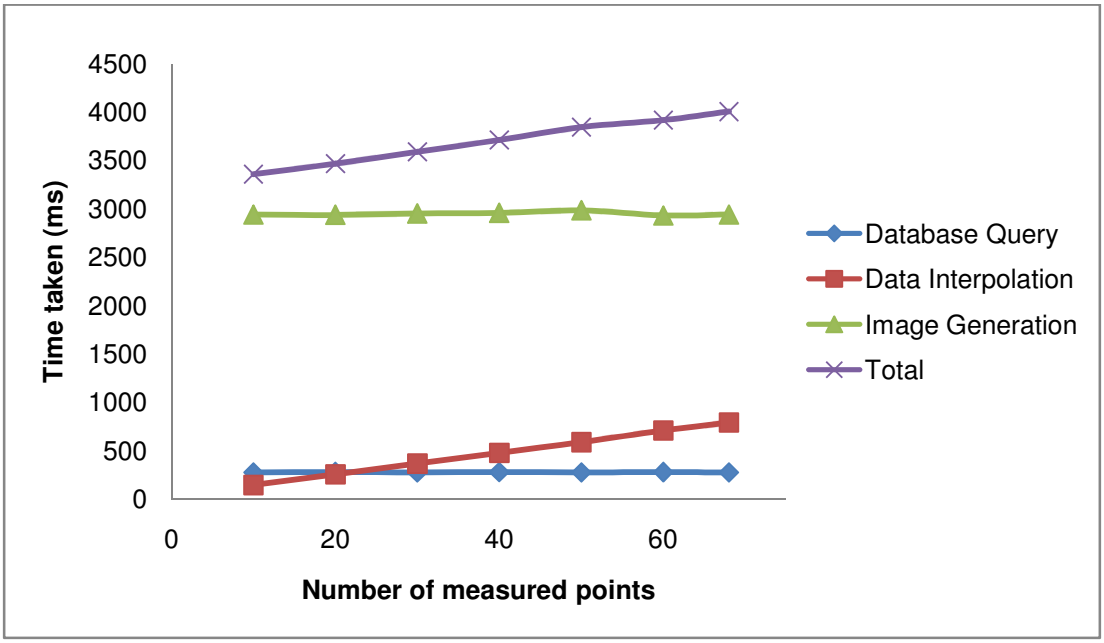


FIGURE 20: TIME TAKEN TO LOAD IDW MODEL

Different time taken versus number of points for kriging

These timings were also carried out on the dataset of CO pollutant values on the 7th of October 2010 at 2 pm. Unlike IDW, kriging does not have a linear relationship between the number of sampled sites and the time taken for data interpolation. It appears to flatten out after datasets go beyond 50. However, this sample size is still too small to be able to confirm this correlation. It should be noted that the recorded time includes the time taken to auto-fit the model variogram, which could explain why data interpolation is faster when the number of sampled sites is small. There will be fewer distance pairs to be calculated and fitted, so processing time will be shorter.

TABLE 6: BREAKDOWN OF TIMINGS FOR DIFFERENT COMPONENTS OF KRIGING MODEL

No. of measured points	Time Taken (ms)			
	Database query	Data interpolation	Image generation	Total
10	276.58	1168.31	2935.26	4380.15
20	277.50	1201.00	2956.16	4434.67
30	278.88	1284.91	2934.68	4498.47
40	278.31	1295.46	2970.46	4544.23
50	279.07	1315.07	2936.06	4530.20
60	280.22	1314.06	2948.94	4543.22
68	279.52	1312.25	2984.05	4575.82

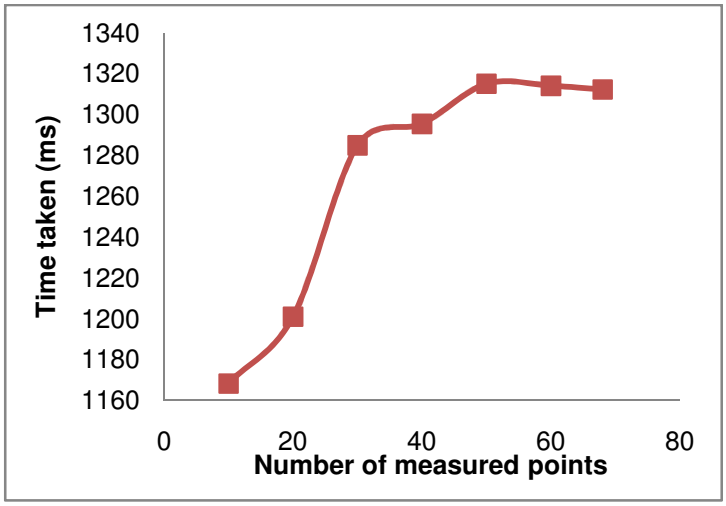


FIGURE 21: TIME TAKEN FOR KRIGING DATA INTERPOLATION

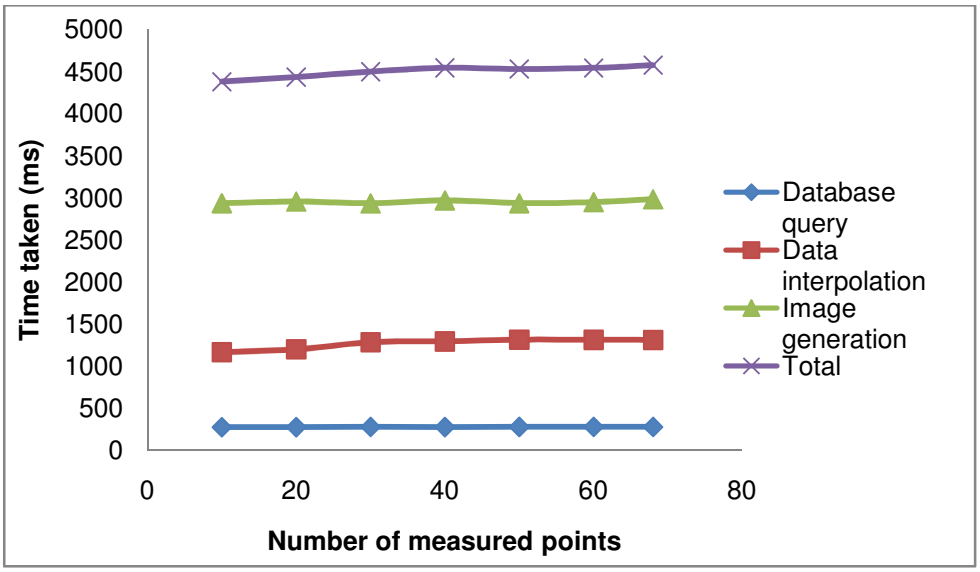


FIGURE 22: TIME TAKEN TO LOAD KRIGING MODEL

When we initialise the model parameters for the variogram rather than using the auto-fit function, the time taken for data interpolation does decrease by roughly 0.2 seconds. With a better quality dataset and a properly pre-fitted model variogram, those milliseconds could be removed.

IDW and kriging comparison

Figure 24 indicates that the IDW system is approximately 600 ms faster than kriging, but this gap closes as the number of measured points increases. Taking purely the data interpolation time itself, IDW is up to 85% quicker than kriging. Even with consideration of the bottleneck of contour generation, IDW is at least 12-20% faster than kriging in total.

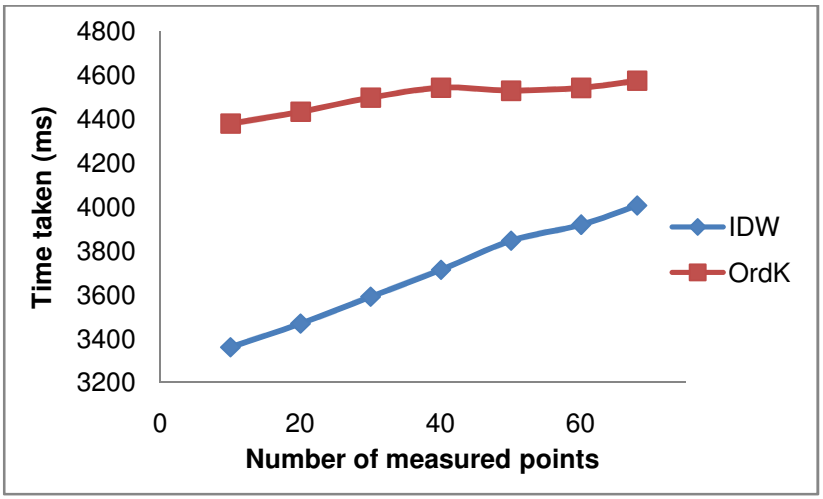


FIGURE 23: COMPARISON OF LOAD TIMES BETWEEN IDW AND KRIGING

It will be interesting to document if there is a certain number of points after which the kriging algorithm will overtake IDW in terms of speed, since IDW continues to scale linearly, while kriging may flatten out.

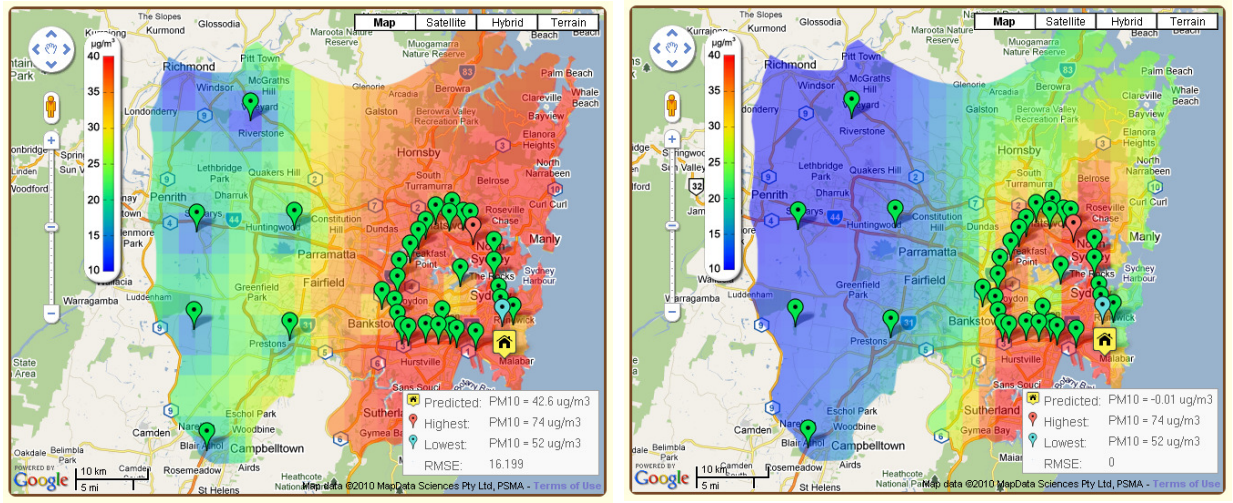


FIGURE 24: CONTOUR MAP FOR PM10 USING IDW (LEFT) AND KRIGING (RIGHT)

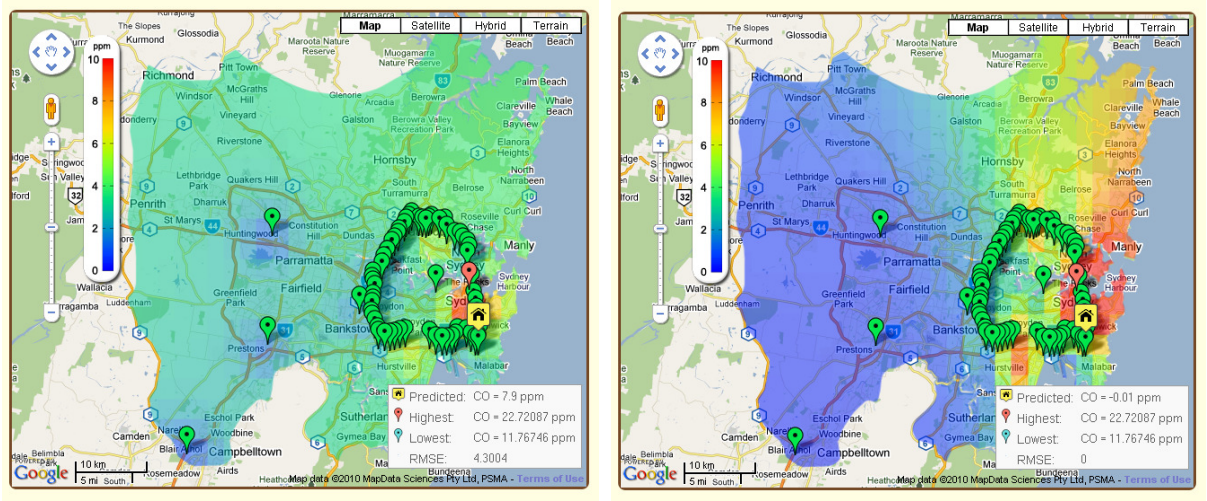


FIGURE 25: CONTOUR MAP FOR CO USING IDW (LEFT) AND KRIGING (RIGHT)

The graphics comparison between the IDW contour map and kriging contour map (Figure 24) show some stark differences. For kriging, the unsafe PM10 pollutant levels are estimated to occur along the associated markers. IDW predicts that all of eastern Sydney is under threat from high concentrations of PM10. On the other hand, kriging estimates that the Sydney harbour is at a high risk of CO pollution (Figure 25), with the rest of the city at safe levels, while the IDW interpolator gives a middle estimate. Kriging does present a much smoother gradient and better aesthetics, compared to the bull's eye effect of inverse distance weighting.

4.2.3 Image generation

The main cause of the long time taken for the image generation component is the size of the image being created. For a 400 x 400 pixel size PNG file, it takes only 0.8 seconds, but takes about 3.0 seconds for the 800 x 800 pixel size PNG file. While it is possible to reduce the size of the generated contour map, this results in a poorer file resolution. There is unwanted increased pixellation when viewing the web application on a high resolution screen. Hence the decision is to maintain the image size at the expense of speed, even as it goes against our objective of having a fast and responsive web application.

4.2.4 Other considerations

The data modelling systems have been built on the back of CGI scripts that call external programs to carry out statistical analysis and plot diagrams. This means that there is a heavy reliance on text files to transmit information between programs. Care must be taken to provide either shared locks or exclusive locks to prevent overwriting of data when it is being accessed by separate applications. This is exceedingly pertinent in the future wherein the likelihood of concurrent processes increases, which will potentially create file conflicts.

Conclusions

5.1 Future work

Reliability of the models is likely to increase when the envisioned network of mobile pollution sensors are set up all around Sydney. An in-depth analysis of the data retrieved from mobile pollution sensors regarding their spatial properties would require a sound grasp of statistics, but would greatly enhance understanding of the underlying assumptions and limits of the data interpolation system. One of the most important tasks would be to carry out heavy field testing in conjunction with the pollution exposure iPhone application. Using a reference mobile sensor device that measures pollutant values without uploading them to the database, we can then compare the actual difference between the sample point and the estimated point to calculate the estimation variance, thereby validating the model and confirming its reliability.

Work could be done in setting up a temporal model to take into account the time lags between the requested time and the time the site was measured. Serious thought must be given to the necessary dispersion models and wind factors, as well as topographic elements. This is especially so for Sydney, where strong winds and multiple hilly areas make it hard to generalise air pollutants over the entire city.

More user interface improvements could be accomplished, such as implementing an overall air quality index (AQI) which would factor in the various air pollutants to derive a single easy-to-quantify value that sums up the condition of air in Sydney. Given that many components of the current system were initially built for desktop applications, they could be further optimised for speed to decrease the loading time for the web tool.

5.2 Final words

When this thesis was first started, there was no apparent solution for the system design but there is now a system architecture in place. Thus we have succeeded in our initial aims of building an effective working web application to estimate and visually demonstrate the air pollution levels in Sydney. There are many improvements that could be made to enhance the quality of information analysis, especially with regards to accuracy. Each additional functionality would increase the worth of the web tool. Even so, this project has helped contribute to a greater understanding of the current state of air pollution and allows us to make better decisions about our personal health risks and how to manage them.

Bibliography

[1] V. Mishra. (2003, Dec.) Health Effects of Air Pollution. Document. [Online] Available: http://www.populationenvironmentresearch.org/papers/Mishra.pdf

[2] World Health Organisation. (2008, Aug.) "Air Quality and Health." World Health Organisation. [Online] Available: http://www.who.int/mediacentre/factsheets/fs313/en/index.html

[3] New South Wales Health. (2009, Mar. 31) "Air Pollution and Health: Key Facts for the Media." New South Wales Health. [Online] Available: http://www.health.nsw.gov.au/PublicHealth/environment/air/media.asp

[4] P. Pulusan. (2004) "Internet GIS -- One Perspective." GIS Development. [Online] Available: http://www.gisdevelopment.net/technology/gis/techgi0035.htm

[5] Haze Watch. (2010) "Haze Watch." University of New South Wales. [Online] Available: http://pollution.ee.unsw.edu.au/

[6] Centre for Scientific Computing. (2010) "CamMobSens." University of Cambridge. [Online] Available: http://www.escience.cam.ac.uk/mobiledata/

[7] Institute for Software Integrated Systems. (2008) "Mobile Air Quality Monitoring Network." Vanderbilt University. [Online] Available: http://www.isis.vanderbilt.edu/projects/maqumon

[8] J. Carrapetta, "Haze Watch: Design of A Wireless Sensor Board for Measuring Air Pollution," University of New South Wales BE Thesis, 2010.

[9] N. Youdale, "Haze Watch: Database Server and Mobile Applications for Measuring and Evaluating Air Pollution Exposure," University of New South Wales BE Thesis, 2010.

[10] Department of Environment, Climate Change and Water NSW. (2010, May) "Air Quality Index (AQI) Values." DECC NSW. [Online] Available: http://www.environment.nsw.gov.au/AQMS/aqi.htm

[11] Lifemapper. (2010) "Lifemapper Services." Lifemapper. [Online] Available: http://www.lifemapper.org/services/

[12] National Center for Geographic Information Analysis. (1990) "NCGIA Core Curriculum in

GIS." University of California, Santa Barbara. [Online] Available: http://www.geog.ubc.ca/courses/klink/gis.notes/ncgia/u01.html

[13] Gamma Design Software. (2010) Gamma Design. [Online] Available: http://www.gammadesign.com/

[14] Golden Software, Inc. (2010) Surfer Product Information. [Online] Available: http://www.goldensoftware.com/products/surfer/surfer.shtml

[15] GRASS Development Team. (2010) GRASS GIS. [Online] Available: http://grass.fbk.eu/

[16] Quantum GIS. (2010) About QGIS. [Online] Available: http://www.qgis.org/en/about-qgis.html

[17] Z. R. Peng and M. H. Tsou, *Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Networks*, 1st ed. USA: John Wiley, 2003.

[18] ESRI. (2010) "Web Mapping Application." ESRI. [Online] Available: http://www.esri.com/software/arcgis/arcgisserver/live-user-sites.html

[19] GeoServer. (2010) "GeoServer." GeoServer. [Online] Available: http://geoserver.org/display/GEOS/Welcome

[20] M. de Smith, M. Goodchild, and B. Longley. (2009) "Gridding and Interpolation Methods." Geospatial Analysis. [Online] Available: http://www.spatialanalysisonline.com/output/html/Griddingandinterpolationmethods.html

[21] G. Bohling, "Kriging," Kansas Geological Survey, 2005.

[22] Yasrebi, "Evaluation and Comparison of Ordinary Kriging and Inverse Distance Weighting Methods for Prediction of Spatial Variability of some Soil Chemical Parameters," *Research Journal of Biological Sciences*, vol. 4, no. 1, pp. 93-102, 2009, http://medwelljournals.com/fulltext/?doi=rjbsci.2009.93.102.

[23] I. Clarke. (2010) Practical Geostatistics 1979. [Online] Available: http://www.kriging.com/PG1979/Chapter_1/index.htm

[24] Coastal Services Center. (2009) "Benthic Habita Mapping -- Spatial Analysis." National Oceanic and Atmospheric Adminstration. [Online] Available:

http://www.csc.noaa.gov/benthic/mapping/analyzing/spatial.htm

[25] Golden Software, Inc. (2002) "Variogram Tutorial." Golden Software Surfer 9. [Online] Available: http://www.goldensoftware.com/variogramTutorial.pdf

[26] Y. Zhukov. (2010, Jan. 16) "Applied Spatial Statistics in R, Section 5." Geostatistics. [Online] Available: http://www.people.fas.harvard.edu/~zhukov/Spatial5.pdf

[27] Google Maps. (2010, May) "Google Maps Javascript API V3 Reference"." Google. [Online] Available: http://code.google.com/apis/maps/documentation/javascript/reference.html

[28] W3Schools. (2010) "AJAX Introduction." W3Schools. [Online] Available: http://www.w3schools.com/ajax/ajax_intro.asp

[29] The jQuery Project. (2010) jQuery UI. [Online] Available: http://jqueryui.com/

[30] The R Project. (2010) The R Project for Statistical Computing. [Online] Available: http://www.r-project.org/

[31] E. J. Pebesma. (2001, May 29) "gstat User's Manual." gstat. [Online] Available: http://www.gstat.org/gstat.pdf

[32] P. Hiemstra. (2004, May 4) "Package 'automap'." CRAN - package automap. [Online] Available: http://cran.r-project.org/web/packages/automap/automap.pdf

[33] GIS Lounge. (2000, Jan.) "Geodatabases Explored -- Vector and Raster Data." GIS Lounge. [Online] Available: http://gislounge.com/geodatabases-explored-vector-and-raster-data/

[34] gnuplot. (2010, Sep.) gnuplot homepage. [Online] Available: http://www.gnuplot.info/

[35] World Health Organisation. (2008, Aug.) "Air quality and health." World Health Organisation. [Online] Available: http://www.who.int/mediacentre/factsheets/fs313/en/index.html

[36] The PHP Group. (2010) PHP: GD - Manual. [Online] Available: http://php.net/manual/en/book.image.php

[37] StatCounter Global Stats. (2010, Sep.) Top 5 Browsers from Sep 09 to Sep 10. [Online] Available: http://gs.statcounter.com/

[38] S. Rahmatizadeh, M. Mesgari, and S. Motesaddi. (2006) "Air Pollution Monitoring with Geostatistical Analysis." GIS Development. [Online] Available: http://www.gisdevelopment.net/proceedings/mapindia/2006/environment%20and%20forestry/mi06envi_148.htm

[39] M. Tomczak, "Spatial Interpolation and its Uncertainty using Automated Anistropic IDW," *Journal of Geographic Information and Decision Analysis*, vol. 2, no. 2, pp. 18-30, 1998.

[40] ArcGIS. (2010) "Performing Cross-validation and Validation." ArcGIS Desktop Help. [Online] Available: http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Performing_cross_validation_and_validation/003100000059000000/

Appendix

Appendix A: Installation Guide

For a Ubuntu setup -

1. Download jQueryUI (UI Core, Interactions and Widgets) from:
   http://jqueryui.com/download

2. Install gnuplot by using the command:
   sudo apt-get install gnuplot

3. Install R by using the command:
   sudo apt-get install r-base-core r-recommended
   sudo apt-get install r-base-dev

4. Within the R environment, update packages before getting *sp, gstat* and *automap*. *sp* provides classes and methods for spatial data, *gstat* handles the kriging modelling and prediction functions, while *automap* automatically fits models. *rgdal* may be required to support the sp package:
   update.packages()
   install.packages('gstat')
   install.packages('rgdal')
   install.packages('automap')
   install.packages('sp')

Appendix B: File Checkout

File Name	Function
public_html	
form-calculate.php	Calculates and prints the estimated point and its variance.
form-contourmap.php	Creates a grid of points and generates the contour map for the entire grid, then masking it with to create a png image.
form-getdb.php	Produces an XML file to store database query results. Is read by the Javascript file to create Google map markers.
form-rmse.php	Calculates the standard deviation of the sampled data and the root-mean-square error of the estimated data. Finds the normalised RMSE with the ratio of std to RMSE.
form-validate.php	Re-validates the form in case of malicious queries. Is used in all the form PHP files.
map.php	Contains the basic HTML code for the layout of the map canvas and form elements.
public_html/cgi-bin	
contour.cgi	Opens a pipe to gnuplot and sends relevant instructions regarding range, colour palette and size. Uses plotdata.txt.
dbdata.csv	Is written over each time a query is retrieved; stores the query results in text form (latitude, longitude and pollutant value).
kriging.R	Creates a grid of points and calls the autofit function, then predicts the kriging estimators. Writes to pollutant_R.txt.
krigingR.cgi	Takes in dbdata.csv, adjusts the grid parameters and then opens a pipe to R, before calling kriging.R. It then reformats the output into a form suitable for graphing in gnuplot, thereby changing plotdata.txt.
plotdata.txt	Is written to by query_process.php and krigingR.cgi. Contains the values to be plotted in gnuplot.

pollutant_R.txt	Is written to by kriging.R. Contains the predicted values and variances obtained from R.
public_html/css	
jquery-ui-1.8.4.custom.css	Stylises the jQuery widgets (datepicker and slider bar).
mapstyle.css	Configures appearance for map.php.
public_html/models/identity	
image_gen.php	Contains the image generation and alteration functions.
model.php	Acts as a shared interface for both the web application and the iPhone pollution exposure app. Contains fundamental functions for system to work.
query_process.php	Contains functions to store and process the query results; from XML and array creation to data interpolation algorithms.
query_todb.php	Parses parameters into specific database queries.