Visualising GPX Data

Student Name: (
Supervisor Nam)

Submitted as part of the degree of **Software Engineering** to the

Board of Examiners in the Department of Computer Science, Durham University.

Abstract –

Context/Background

The purpose of this paper is to answer the question of whether a piece of software can be constructed that can output a two-dimensional visualisation based on the input of a GPX file.

Aims

This project aims to take the main research areas of information visualisation, as described by Spence & Chen, to produce an output that provides the most use to the user of the data with an input of a GPX file.

Methods

For the creation of the system version 1.6.0 of the Java programming language developed by Sun Microsystems was used. A GPX parser was implemented using Document Object Model (DOM) and JXMapKit was used to display an instantiated version of OpenStreetMaps, an open source map. Various Java packages and code fragments were used to post-process the map and display the waypoints parsed from the GPX document.

Results

The final system meets the points raised above and provides the user with a tool for viewing GPX data on an interactive map. The system uses many features of a good visualisation as discussed by Spence and Chen, mainly the importance of ensuring that all visualisations provide useful information for the user which includes the removal of redundant data from the visualisation.

Conclusions

This paper describes the creation of a tool that meets the expected results and a number of visualisation techniques that it implements.

Keywords **–** JXMapKit, Java, GPX, Mapping, Parsing, DOM, OpenStreetMaps, Visualisation.

I. Introduction

Cycling and walking have always been popular pastimes amongst a large proportion of the population. These pastimes have become the catalyst for the rise in production and sale of satellite navigation and tracking devices that use the Global Positioning System (GPS) (ABI Research, 2007). The use of these devices has escalated since the year 2000 when the US military, who own the GPS satellite network, allowed the general public to receive unencrypted GPS signals and therefore have access to accurate location data.

Modern GPS devices have the capability to create and load a special file that holds raw data clusters called ‘waypoints’ that contain at least latitude and longitude but also other data such as elevation for a certain geographical point. A problem that comes with using these devices

is how best to display the raw data clusters as information so that the user gains full understanding of the contained data. This raw data format is named GPX or the **GPS eXchange Format**.

A GPX file can be created in one of two ways. GPS devices can be setup by the user to take a "snapshot" of information containing the latitude, longitude, elevation and time at a particular point at a specified interval. This allows the user to walk or cycle down a particular route and at every defined interval (say, 1 minute) the GPS device will capture this position information and append it to the end of the GPX file. When the route has been completed the user can stop the GPS device recording. When the user returns to their computer they can upload the newly created GPX file via USB interface.

The second method of creating GPX files involves using an existing piece of software that allows the user to specify the latitude and longitude of a certain point and the software will calculate the elevation data from a look-up table. These waypoints are added in sequence and the software allows the output of a GPX file.

Several solutions currently exist for displaying GPX data but many are often applications created by hobbyists and lack good design principles, stability and functionality that is useful for the user. A good solution will adhere to the points raised in the Keys to Success section.

A. Project Aims

This project aims to create a solution to the problem of how to display the data stored in GPX files to the user in a useful manner using several design principles for the creation of a good visualisation. As a research question the problem can rewritten as: *"Is it possible to create a piece of software that can take GPX data as input and produce an output that is useful for the user?"*

This project proposes a solution to the problem of how to display GPX data using suitable elements from the vast topic of information visualisation. This methodology combines modern data structures and types with traditional visualisation techniques developed from the 1980's through to modern day.

The overarching scheme of this project is to convert complex raw data into a user-understandable format in the form of a visualisation. The phrase "a picture is worth a thousand words" holds true in the principle of this project. Psychologists have amply demonstrated the advantages of imagery over numbers for certain applications (Shneiderman, 1980). In this case the quantity of raw data held in GPX files would be far too time-consuming to comprehend and therefore useless for a user who would like to understand a certain route. It is for this reason that a visualisation is the most suitable means to convey meaning of the data.

B. Motivations for Project

Both GPS technology and the study of information visualisation are relatively new areas of research in the computer science field. With GPS recently reaching a maturity level, in terms of market saturation and technological advancement, value has been added to visualising the output data.

In a practical example, by overlaying the GPX data onto a map (one type of visualisation) a user could explain walking routes clearly and accurately to a friend. Using the time and date data available, the user is also given an idea of time scale for a route. This could be extended further to allow joggers and runners to set a completion time for a route and other users could match this time or try and it beat it as part of a training schedule.

C. Keys to Success

The task of visualising GPX data poses a challenge due to the vast data elements that can be stored within a GPX file. This task is made harder by the average data retention of the short term memory of a human being. Miller (1956) found remarkable evidence that the channel capacity of a human is equivalent to between 5 and 9 equally-weighted errorless choices. Or simply, the average human can remember and understand 7 facts (plus or minus 2) in their short term memory. This is a very important notion to adhere to when designing a visualisation project.

In one visualisation window on the screen there should be at most 9 items of information. Any more and the information is greatly devalued for the user. As GPX files can contain any number of data a decision has to be made on the most important types to be persistently displayed. The previous point and the decision of how the data should be displayed are the two most important decisions to ensure that this project is a success.

Many forms of information visualisation exist from TreeMaps (see Related Work) to simple graphs and a key issue with this project is to select the most appropriate visualisation for displaying GPX data.

II. Related Work

A. GPX File Format

The "GPS Exchange Format" file format is a data standard that defines how geographical mapping data should be stored (Topografix, 2008). Each geographical point (cluster) that is required to be stored is categorised into 1 of 3 interchangeable headings depending on the device used for output: "Waypoint" (the most common), "Trackpoint" or "Routepoint." Each of these headings can contain any additional data that adheres to the GPX specification.

The GPX format is an XML schema and therefore exhibits all of the properties of XML such as nested tags and a hierarchical structure which is necessary for storing geographical data.

```
<trkpt lat="54.454080" lon="-3.317148">
<ele>399.08</ele>
<time>2006-06-03T13:43:40Z</time>
</trkpt>
<trkpt lat="54.454100" lon="-3.317299">
<ele>403.41</ele>
<time>2006-06-03T13:44:38Z</time>
</trkpt>
```

Figure 1. An example GPX file. Each cluster is categorised as a Trackpoint ("trkpt") containing Latitude and Longitude information (coloured red), elevation information (coloured blue) and time information (coloured green) for each geographic point.

Figure 1 shows an example of the data found in a GPX file. This example holds data about the elevation at each point as well as the current time when each point was visited.

GPX files are classed as a hypervariate data source because each cluster may contain more than 1 attribute. Figure 1, for example, may hold some more attributes than the ones displayed

or less. This poses a problem as hypervariate data is difficult to display all at once and still be comprehendible. Therefore care must taken to ensure the notion of high importance of intuitive and understandable visual patterns are upheld (Müller, 2008).

The standards created for GPX files are upheld by an organisation called Topografix (2008) and the latest version is 1.1. Version 1.1 is a significant improvement over version 1.0 for many reasons. Firstly it allows for up to 3 distinct clusters of information (Waypoint, Trackpoint and Routepoint) which is useful for mapping different routes on one visualisation. Most importantly, version 1.1 defines how the latitude and longitude information should be stored in the file. Up to version 1.1 it was common to see these data items held in either a tag, and element or an attribute (Figure 2).

```
<trkpt lat="54.4304881685" lon="-2.9710207335"><ele>45</ele></trkpt>
<trkpt lat="54.4323351208" lon="-2.9701181701"><ele>44</ele></trkpt>
<trkpt lat="54.4325435788" lon="-2.9700161624"><ele>44</ele></trkpt>
```

(a)

```
<wpt>
 <lat>42.439227</lat>
 <lon>-71.119689</lon>
 <ele>57.607200</ele>
</wpt>
```

(b)

Figure 2. An extract from a pre-version 1.1 GPX file. Note the various ways that latitude and longitude data is stored. (a) shows the data stored as attributes, (b) shows the data stored as elements.

B. Information Visualisation

A large amount of research in the area of Information Visualisation has been done by Spence (2001) and is presented in the book "Information Visualisation." In this book Spence describes eight categories that all visualisations should encompass:

1. Selection – suitable data should be selected for a particular task. Can this selection take place automatically? Is it useful sometimes to suppress information? Is selection very fundamental to information visualisation?
2. Representation – data should be encoded in some way for presentation to the user. Many methods of encoding are possible. Which are useful? Can they be combined?
3. Presentation – Each author has to lay out their data in some way. Playfair (Biderman, 1990) lined up the empires so that the differences in slope were obvious. Often we find we have more data than can easily be displayed at once. What can we do about this presentation problem?
4. Scale and Dimensionality – the designer of the visualisation must be aware of ways in which scale influences the way in which visualisation tools are designed. Another consideration is the dimensionality of displayed data (how many features can be incorporated?) Playfair considered two features and Minard (Tufte, 1983) more.
5. Rearrangement, interaction and exploration – The ability to explore data by rearranging it interactively is so valuable that a great deal of effort has been invested in the invention and implementation of interactive implementation tools that harness this potential.
6. Externalisation – the concept of visualisation refers to an internal model in the mind of the user. What the user sees on a computer display is called the externalisation of the data. The way in which data is externalised is crucial to the success with which visualisation is achieved.

7. Mental models – visualisation is essentially a human activity, if we can understand how an internal model works then we are well placed to design a visualisation tool.
8. Invention, experience and skill – all visualisation tools have to be designed. This is a task that is complicated in view of the complexity and unpredictability of typical tasks, the fast changing palette of interaction techniques available to the designer, and our lack of human-computer interaction in general. (Spence, 2001).

For this project the most important categories are selection and representation as raw GPX data contains many elements that can be displayed on a screen from latitude and longitude to elevation. On their own these elements are difficult to read and interpret as they are part of many other elements in a flat XML based file.

One of the largest challenges in information visualisation is how to display data with more than one variable. The data contained in GPX files is an example of hypervariate data. One solution to displaying hypervariate data is through the use of parallel coordinates (Figure 3) (Inselburg, 1985; Wegman, 1990).

The principle behind parallel coordinates involves taking all of the axes of multidimensional data and rearranging them so that they are parallel to each other. An example of a common use of parallel coordinate plots is plotting financial data which usually contains many variables.

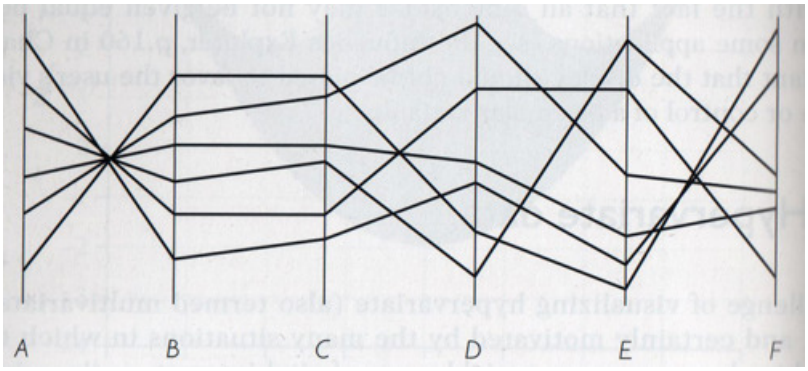


Figure 3. The theory of parallel coordinates plots applied to six variables (Spence, 2001).

Malone (1981) has noted that human beings have excellent spatial memory, a characteristic that can be exploited in many ways in the field of visualisation. This skill lends to the idea of displaying GPX data as an overlay of the map of the World.

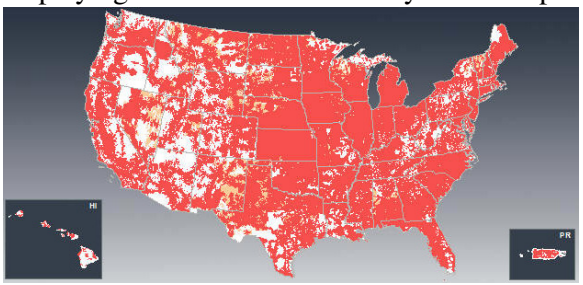


Figure 4. Map showing the wireless coverage of the Verizon mobile network in the USA. (Verizon, 2008).

Figure 4 is a good example of spatial data representation as it shows a map of the USA that many people can understand due to the high degree of spatial memory present in human beings. The red overlay provides data about the coverage of a mobile network.

Shneiderman (1992) describes how information that has multiple paths can be displayed in

way that maintains the overall path integrity. When any data that is collected shows connectedness and interrelationship, a tree data structure can be drawn. An alternative representation of a tree data structure is the TreeMap. Shneiderman describes the construction of a TreeMap:

"*Starting with the designated root node, one draws a rectangle…Within the rectangle are smaller rectangles, one for each subordinate node of the node just considered. This construction is repeated until all nodes are accounted for.*" An example of this is shown in Figure 5.

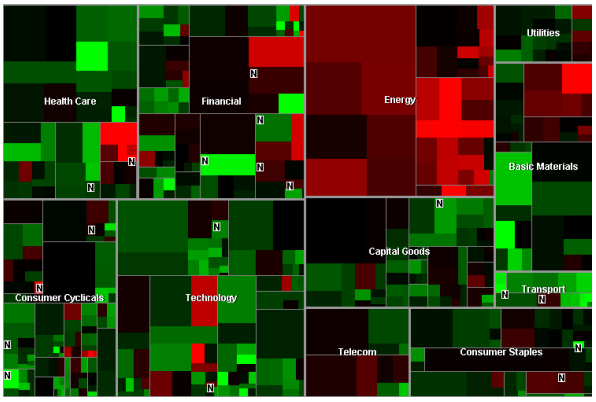
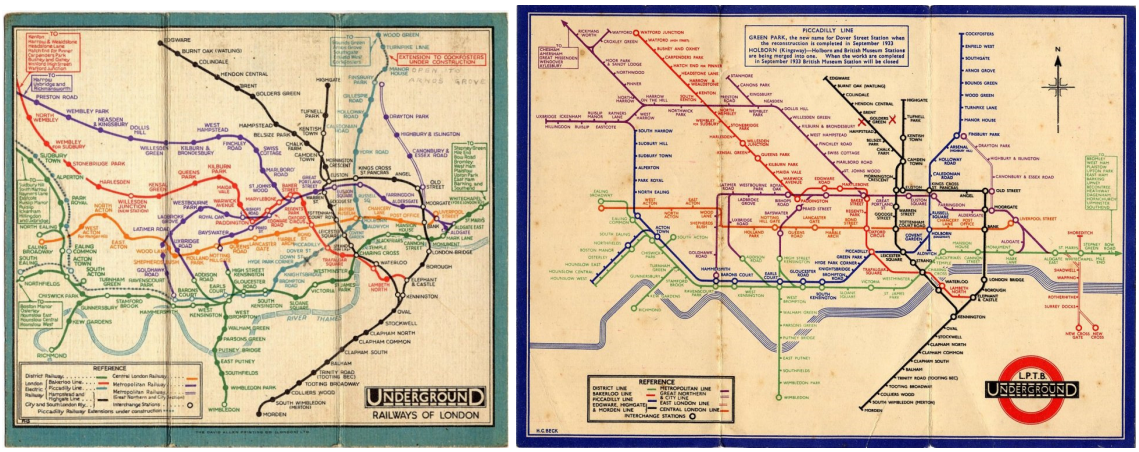


Figure 5. A TreeMap showing the distribution of monetary investment in the Dow Jones (Smart Money, 2008).

Although this visualisation is useful for certain data such as financial or hierarchical data with a large number of tiers it is not suitable for GPX data files as they only contain the data for single clusters of waypoints with many variables, which are not complex and large enough to lend themselves well to a TreeMap. This example has been included for comparison of different visualisation types and the suitability of some compared to others.

An example of a visualisation that has similar properties to this project is the contemporary map of the London Underground transportation system designed by Harry Beck (Garland, 1994) (Figure 6). This redesigned map provides information about many routes that can be travelled on using the 'Tube' but the information is distorted and abstracted from the real life paths. From an information visualisation aspect the contemporary map removes confusion and makes it easier to gain an understanding of current position and required routes by redesigning every route as a straight line with the "orderly precision of an electrical wiring system" (Bryson, 1998). The Underground map provides the reader with a 'cognitive map' which is an internal memory model holding simple but vital information that has been abstracted from the original arrangement. For example if a commuter wishes to travel from King's Cross to Old Street the new map allows them to remember the journey as simply as: "Black line south; 2 stops."

Therefore according to Spence's rules for information visualisation, the redesigned system is superior compared to the old map and of more use to the user. It is this abstraction that makes the redesigned Underground map a very good example of a visualisation.



(a) (b)

Figure 6. Two versions of the London Underground map. (a) The pre-Beck map. (b) The improved map created by Harry Beck.

C. JXMapKit

JXMapKit is part of a series of libraries created by Java SwingLabs (2008) that can connect and retrieve data from web services. The JXMapKit itself is a generic viewer for tile based map servers and it allows the connection to several online maps and the download of individual tile data. The front end of this system is called JXMapViewer and it provides the ability to scale, pan and interact with the underlying map tiles (Figure 7).

JXMapKit is a Java library that is compatible with Java version 1.5 and above. The library can connect to a variety of web based map servers including NASA's Blue Marble (2008), and OpenStreetMap.org (2008). This allows the JXMapViewer to present many forms of map from satellite imagery to road maps depending on the requirements of the system.

The main advantage of the JXMapKit for a developer is its ability to implement the Java Painter class for differing purposes. The Java Painter class allows the developer to define shapes and text using a variety of pre-defined shape models and colours. The JXMapKit contains methods to draw a predefined Painter class for every instance of Waypoint. This means that a base shape can be drawn for every waypoint on the map and by tweaking the Painter class a different text element can be drawn for each waypoint. This is a very useful tool for visualising routes on a map.

III. SOLUTION

A. Architecture

The system was created using the Java programming language programmed with the Eclipse development environment. The design of the system can be represented in 3 separate stages:

1. The creation of a GPX parser to allow the data contained in a GPX file to be sorted and stored in memory. This includes several algorithms to check for the many possible data types in the file and store these accordingly as well as reading the GPX version number in the file header to ensure the correct element names are parsed.

2. The implementation of JXMapKit (Figure 7) to provide an interactive map of the world containing street names and other visual indicators. The map kit allows for the ability to zoom in and out and panning.
3. This is the visualisation stage where the data contained in the GPX file is incorporated with the JXMap. This stage contains several algorithms to allow the visualisation to change depending on the available data in the GPX file.

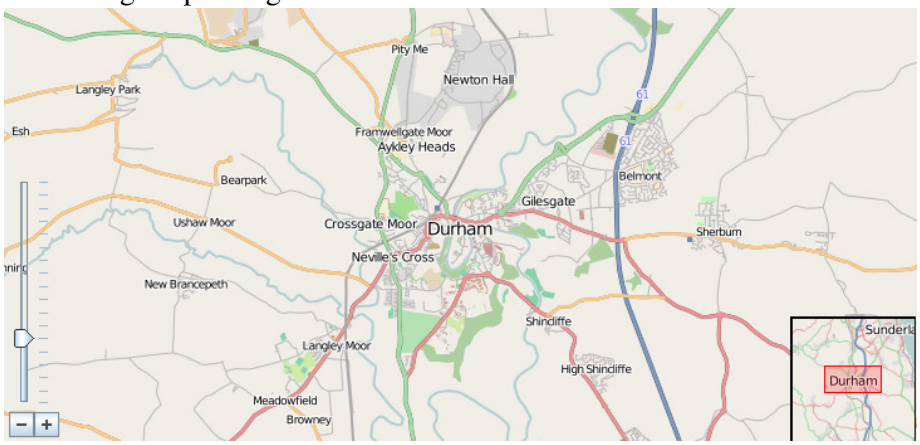


Figure 7. Screenshot of a JXMap implementation. Note the slider on the left to allow for zooming and the box on the right that allows for panning.

In addition to the JXMap to visualise the data, an implementation of JFreeChart (2008) has be used to show the elevation over the route in the GPX file (if elevation data is present in the file) (Figure 8). This part of the system required an algorithm to sort the elevation data and convert it into a data type that JFreeChart can take as an input to display the graph.

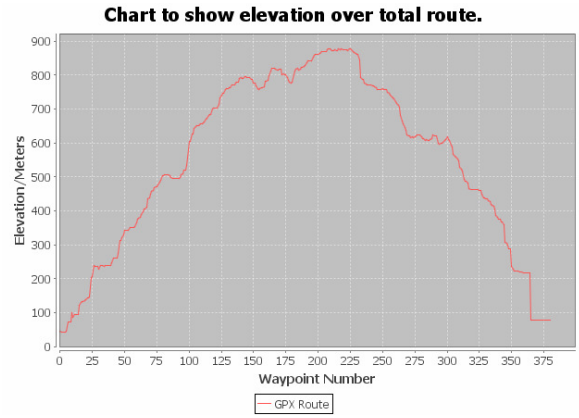


Figure 8. A JFreeChart showing the change in elevation over a route.

JFreeChart is a chart library implemented in Java. It contains a well-documented API which supports a wide range of chart types. The API for this library is simple and therefore the results produced are not that customisable. However, the advantage of using JFreeChart for this project is that it can take as input the labels for the X and Y axis and a Java List of points and it will automatically draw the graph with the correct scaling and presentation. This is very useful for quickly displaying the change of elevation over the course of a route. Further set-up to the library allows the user to right click on the graph and alter colours and scales. The user

can also save the graph as an image for later reference using the same right-click menu.

In terms of the parsing the GPX file two different APIs exist that can be used to parse through standard XML (of which GPX files are a sub-set). The two implementations are SAX (Simple API for XML) (saxproject.org, 2009) and DOM (Document Object Model) (W3C, 2009).

Both have advantages and disadvantages when used for different systems. The main difference between the two is that SAX is a stream parser which means that different events are called when a certain defined node or element is encountered. This allows the developer to define *what* to parse as opposed to DOM which is more explicit in nature and requires defining *how* to parse a document. An advantage of SAX is that is uses a smaller amount of memory during execution when compared to DOM. However DOM has many features that are more valuable for this project than SAX.

DOM provides a detailed API that is constantly updated by the W3C ensuring features and improvements are added regularly. It can also be tailored to parse through unusual XML files such as GPX files. This is especially helpful when the different GPX standards signify that the GPX structure can change. By defining what elements and attributes DOM should look for when parsing makes the system very flexible when it comes to reading in differing GPX files. These advantages outweigh the advantages of the SAX parser and the waypoint data held in a GPX file is usually small (no more that 1000 waypoints per file) so the memory usage advantage with SAX is redundant in this project. It is for these reasons that DOM was chosen as the parser in this system.

B. Design

The 3 elements of the system can be set to execute and process data completely separately therefore they adhere to the object-orientated design of modularisation. Coupling between the modules is kept very lose with a maximum of two methods linking them together. The Parser class is the central class that receives requests from the GUI and sends requests to the Ordering System. This architectural diagram can be seen in Figure 9.

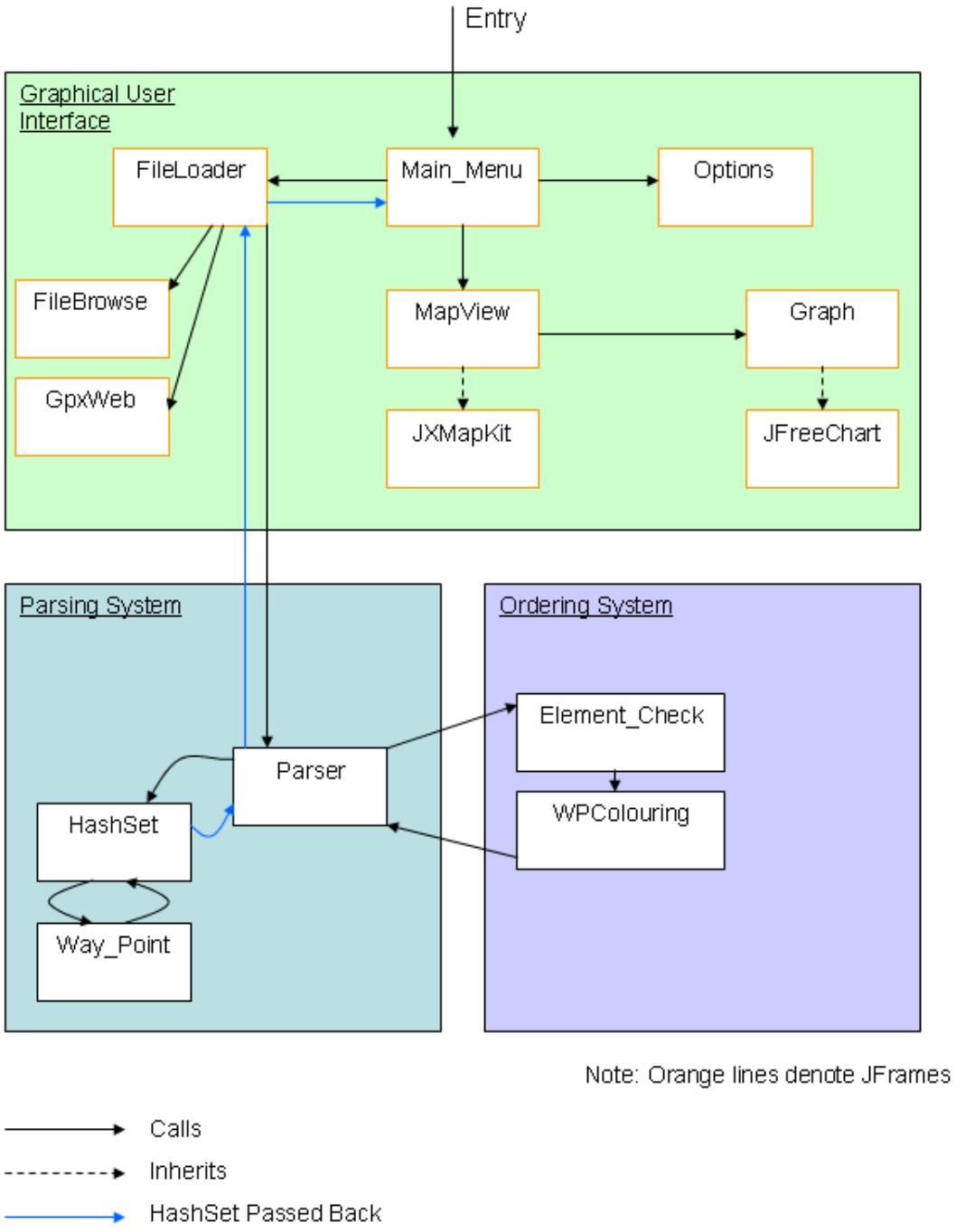


Figure 9. The architectural diagram of the GPX visualisation system.

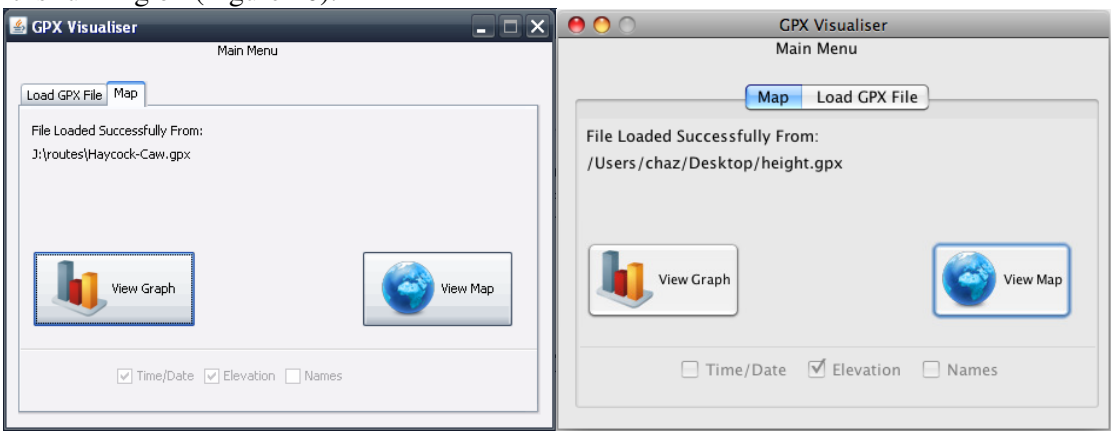
C. User Interface – Main Menu

When designing a user interface it is important to adhere to certain rules. Foley & Wallace (1974) defined 5 cautionary rules for interface design.

1. Boredom – improper pacing
2. Panic – no long delays
3. Frustration – unforgiving systems

4. Confusion – lack of structure or inappropriate detail
5. Discomfort – inappropriate physical environment

As such care has been made to ensure every part of the interface has optimised usability. The GUI has been created to obtain the styling information from the native operating system it is running on (Figure 10).



(a) (b)

Figure 10. The GUI with native operating system window manager support. (a) Windows XP. (b) Mac OS X

The main menu itself has been separated into 2 tabbed areas that contain distinct elements to interact with the system. The "Map" tab allows the user to open the main map and/or elevation graph to view the loaded route and the "Load GPX File" tab allows the user to select and load a GPX file from the file system or a GPX file from a web server. The simple two element layout of the GUI hides the complex class and process interaction and provides the user with a comfortable physical environment to work with which promotes the use of the system. The use of icons on the buttons also adds to the feeling of comfort by providing visual indicators of what the buttons will do when pressed. The two tabbed design of this system allows the user to move from the two ends of the system (loading and viewing) very quickly and therefore reduces boredom and frustration so most of the user's time is spent working with the data and not navigating the interface.

All the text used in the system uses natural language communication instead of short imperative statements, which are unsuitable for non-technical users. Natural phrases such as: "Please Click 'Load GPX File' to continue" instead of "No GPX File Loaded" are used that help to reduce confusion for the user and allows the user to feel more in control of the system.

When a GPX file has been selected from the file system and has been sent to be parsed the user is informed of every step of the parsing (a potentially slow process) through the use of natural language communication. This ensures that there are no seemingly long delays with using the system therefore reducing panic for the user.

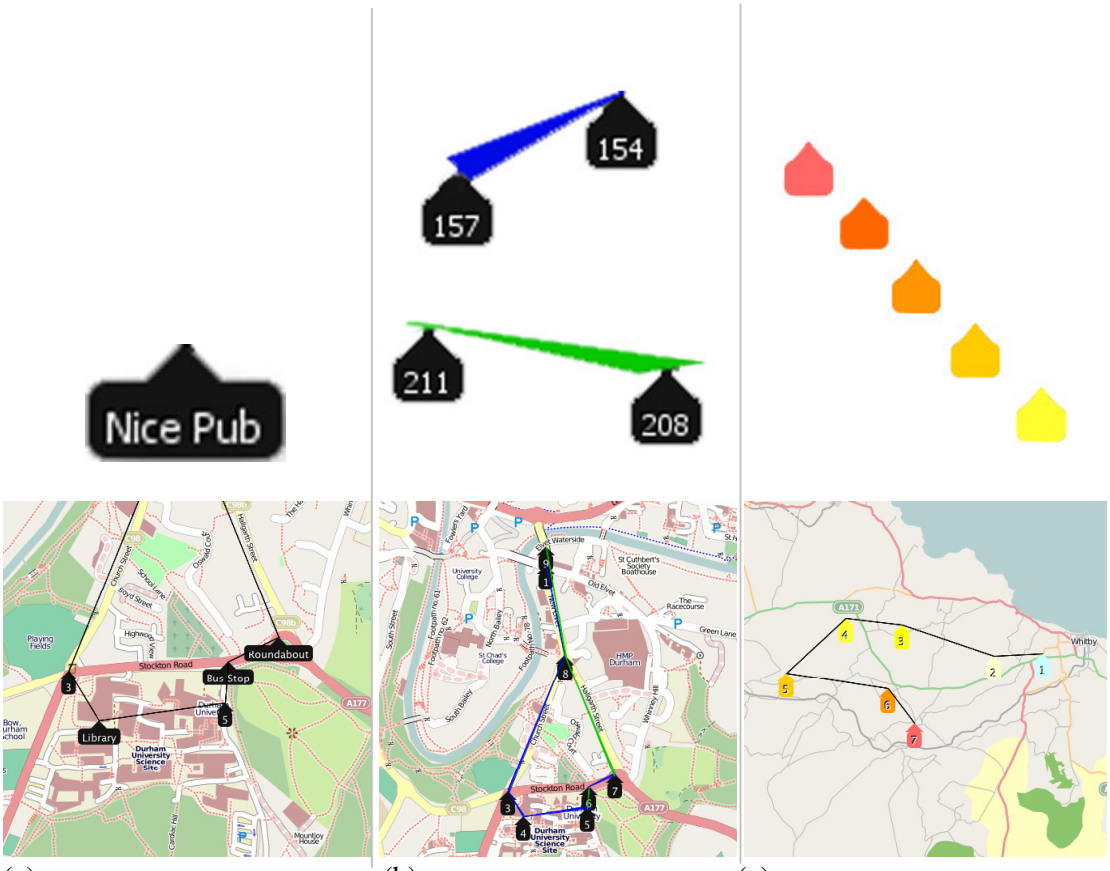
The combination of these above points creates a system that is user-friendly and easy to navigate and removes boredom, panic, confusion and discomfort. Therefore the system is very forgiving and this removes frustration from the user experience.

D. User Interface – Visualisation

When a GPX file has loaded successfully it is ready to be visualised using either the map or graph (if elevation is present).

When the map is opened up the viewport will centre on the first waypoint found from parsing and the zoom will be set to a close level as the majority of routes cover a short distance in comparison to the size of the planet. Depending on the type of data that was found in the GPX file a combination of the 3 defined elements can be shown on the map.

1. Name – Each waypoint can be assigned a name which is shown on the waypoint marker (Figure 11 (a)). This allows the user to differentiate between waypoints and find special waypoints that may define an area of exceptional interest.
2. Elevation – The elevation between two points is denoted using a coloured line. A green line represents a descent (symbolic for grass) and a blue line represents ascent (symbolic for sky). Two points that have no change in elevation between them are shown using a black line to join them. The magnitude of elevation is depicted by the degree of the width of the tip of the rectangle. A blue rectangle showing ascent starts narrow and ends with a larger width. This larger width is dependant on the step-up in elevation. The smallest end width denotes a change of 3 meters or less while the largest width shows a change of 30 meters or more. The same principle is applied to the green descending lines however the thick end of the line is displayed at waypoint A and will reduce in thickness by waypoint B showing the extent of descent.
   An advantage of this method is that it is easy for the user to follow the path of travel from any waypoint in a route because the elevation lines are vectors and show direction as well as magnitude.
3. Time/Date – The presence of Time and Date data is determined by the parser. If it exists then the waypoints are ordered chronologically to show the actual route travelled. For each waypoint the time data is read and compared to a series of values that place the time of day in a predefined bracket. This bracket will classify the colouring of the waypoint. Leland (1998) associated the time of day with different colours from bright orange depicting mid-day and purple for late evening. Figure 11 (c) shows a route that began in the early morning and finished late evening thus showing all possible colour values.



(a) (b) (c)

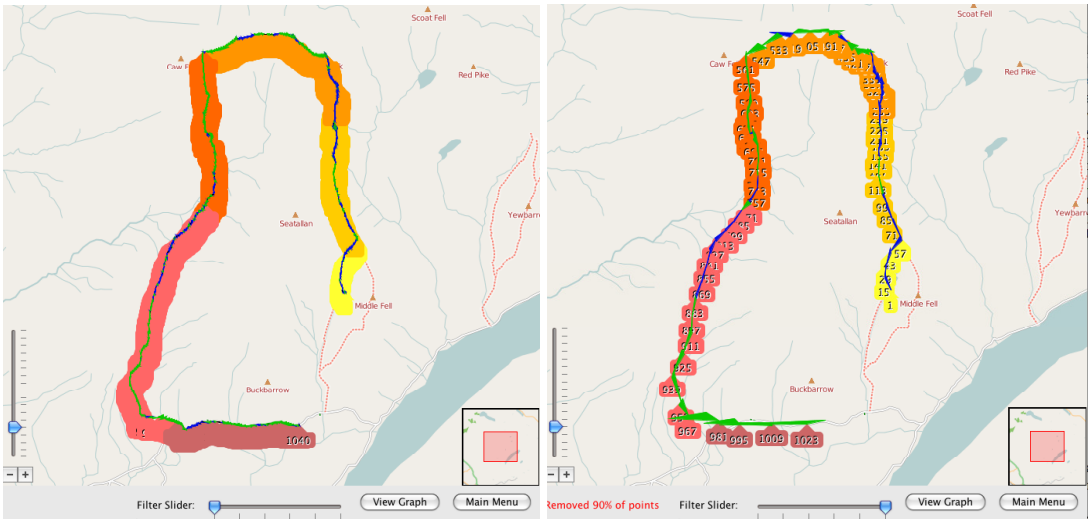
Figure 11. The map viewer showing: (a) Waypoint names with a sample waypoint marker above. (b) Elevation with two sets of sample waypoints with elevation data above. (c) Time/Date with 5 sample waypoint markers showing the change in waypoint colour.

At the bottom of the main map window is a slider with 6 positions labelled “Filter Slider.” In the default position the waypoints will all be displayed on the map as normal. The second position will remove 50% of the waypoints and every further position will remove 10% more up to 90% on position 6.

The filtering system is valuable if a GPX file is being displayed that contains many waypoints over a short distance that are clustered together. The close clustering of waypoints also makes it hard to see the elevation lines if they are present. By filtering the waypoints the path of the route is maintained but the wider spacing of the waypoints allows the user a greater understanding of the elevation changes. Also the underlying map can be seen more clearly due to the lack of occlusion by waypoint markers (Figure 12).

When filtering the elevation of the remaining waypoints are compared to the waypoints that have been removed. An average is formed for the elevation between these points and a new elevation line is drawn between waypoints to reflect this change.

The user is informed if the waypoints are being filtered by a red text label that appears to the left of the filter slider. This label will inform the user that the waypoints are being filtered and by what percentage. This label is important so that the user does not get confused when the waypoints are removed due to filtering.



(a) (b)

Figure 12. The map viewer showing: (a) Unfiltered waypoints. (b) Waypoints filtered to 90%.

If the GPX file contains elevation data the user can also choose to view the change in elevation over a route using the JFreeChart implementation. The graph will show the elevation of all the waypoints regardless of filter level.

In the graph view the user can right click and bring up a menu containing formatting options or send the graph to be output on a printer for later analysis.

IV. Results

Due to the hypervariate nature of GPX files and the many options available in the implementation, the results produced will be varied and numerous. Primarily, it is important to discuss the results that the user will view as the system is built around the notion of visualisation.

The three areas of the system described in III.A have been brought together within one graphical user interface. This GUI contains sections to allow the user to open a GPX file for parsing, view the main map and view the elevation graph (if available). As the design of the system is linear, areas of the GUI are disabled until the precursor step has been completed. For example, the map cannot be viewed until the GPX file has been parsed.

On the main screen of the GUI a series of check boxes inform the user of what data is available in the GPX file (Figure 13). This allows the user to understand what data they will be looking at when the main map is running.

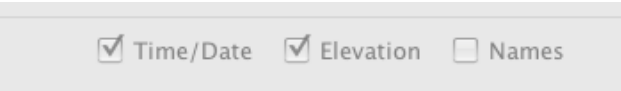
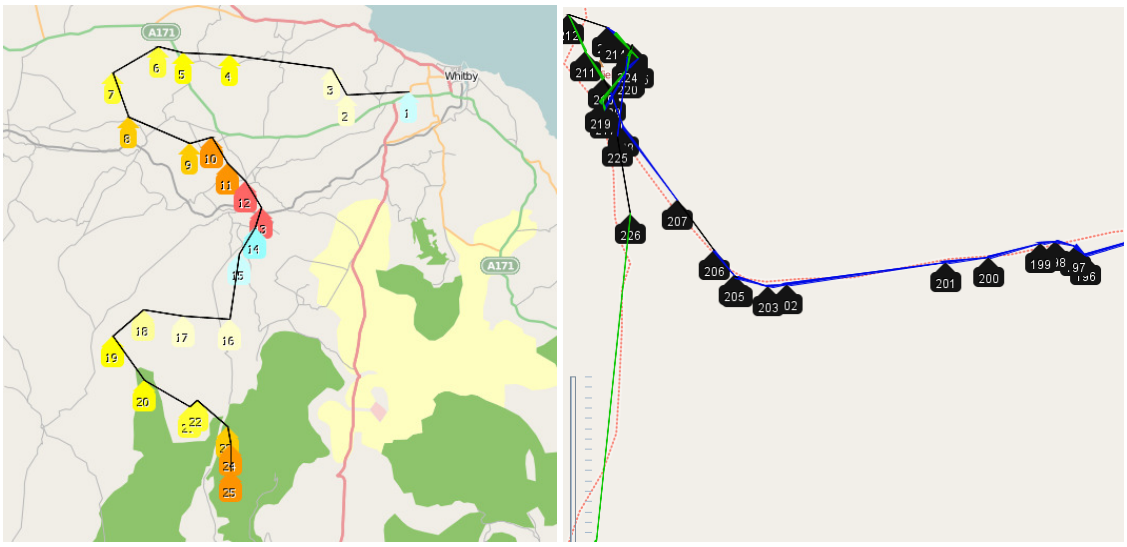


Figure 13. Three check boxes that inform the user of the type of data that is available in the GPX file. This example shows that elevation data and time/date exists but not name data.

The three types of data available (Figure 13) are represented on the JXMap in different ways depending on availability.

1. Time/Date: If the GPX file contains calendar data then the waypoints are ordered by date in the parsing stage using a sort algorithm. This allows the route to be viewed on the map with each waypoint displayed in order and a line connecting adjacent waypoints in the order of the route. For a greater understanding of the time scale involved in a route, each waypoint marker is colour coded with a representation of the time of day. This is a visualisation topic in itself but it is found more in artistry then science and has been discussed by Leland (1998) (Figure 11 (c)). As can be seen the results produced allow the user to obtain a good understanding of the time taken to complete a route. If a route continues for more than one day then this is easily seen on the map as the waypoint colours would recycle (Figure 14 (a)).
2. Elevation: is displayed in the program through both the main map and a graph. In the map a drawing algorithm is used that parses through the waypoints comparing adjacent elevation data. If the elevation decreases from one waypoint to the next then a green line (to represent grass) will be drawn between waypoints and a blue line will be drawn if the elevation increases (to represent sky). The size of the elevation change is shown by a fanning line with the fan width being larger for a greater change in elevation (Figure 14 (b)). If there is no change to elevation then a straight black line is drawn.



(a) (b)

Figure 14. The map viewer showing: (a) a route lasting two days – note the two sets of colour changes. (b) a route with varying elevation changes.

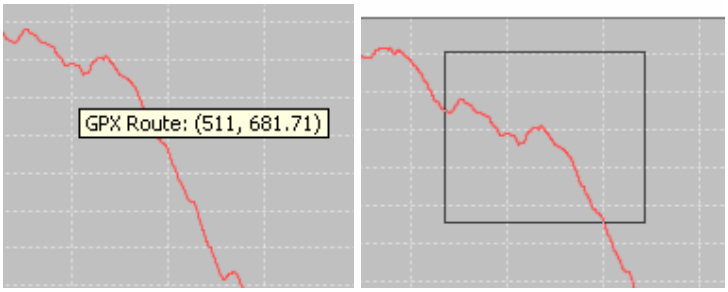
3. Names: If the GPX data holds the names for each waypoint then they are displayed in the individual waypoint markers (Figure11 (a)). If no name information is present then the waypoint markers are numbered in the order that they were travelled through.

If the GPX file does not contain any suitable data or the parser cannot locate the GPX version number in the header file then the system will attempt to find latitude and longitude information to construct a map of the information. If this occurs then the user will be informed in the parsing console.

If there is an error in reading the file or an invalid file has been entered then an error message will appear informing the user in clear natural language and the user will be prompted to find another file.

In the graph window the user can interact with the elevation data in a number of ways. Right-clicking on the graph brings up a context menu that contains several helpful options. The 'properties' sub-menu allows the user to format various aspects of the graph including line colour, axis range and background. When the required format has been achieved the graph can be saved as an image file or output to a printer using the right-click menu.

If the user hovers the mouse pointer over any point on the graph a pop-up box will appear detailing the point number and the elevation at that point (Figure 15 (a)). The user can also zoom into the graph to obtain a better view of a cluster of points. This can be done by clicking and dragging a rectangle frame (Figure 15 (b)), when the button is released the graph will be redrawn using the selected frame as the new bounding axis.



(a) (b)

Figure 15. The elevation graph view showing: (a) waypoint number and elevation in meters for a particular point. (b) the zooming system with a user drawn box that will become the new central focus of the graph.

V. Evaluation

The visual nature of this project does not lend itself well to quantitative analysis as the results are entirely visual and subjective depending on the requirements of the user. As such, a pre-defined framework should be used to evaluate the system in a qualitative way. One such framework is the "Eight Rules of Interface Design" that have been proposed by Shneiderman (1992). These rules were obtained from a book written by Shneiderman and are a collection of principles that have been derived from his personal experience and are applicable to most interactive systems. The points below outline the rules as described by Shneiderman and the sections in italics show how this project meets each of Shneiderman's points:

1. **Strive for consistency.**
   Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent commands should be employed throughout.

Every set of buttons in the program that perform the same action (return to main menu, show/hide graph) are labelled identically to aid the user with understanding the system. The same is true for error messages. Throughout the program "Currently Viewing" labels are placed, which contain the file name of the current GPX file. These labels are written in the same format to remove doubt from the user about which data set is being viewed.

2. **Enable frequent users to use shortcuts.**
   As the frequency of use increases, so do the user's desires to reduce the number of

interactions and to increase the pace of interaction. abbreviations, function keys, hidden commands, and macro facilities are very helpful to an expert user.

N/A – The interface has been designed in a simple way to allow the user to navigate without the use of shortcuts. This has been possible by using two tabs containing options to load a GPX file and view the data.

3. **Offer informative feedback.**
   For every operator action, there should be some system feedback. For frequent and minor actions, the response can be modest, while for infrequent and major actions, the response should be more substantial.

Every step of the GPX file parsing progress is displayed in a text area to allow the user to view the stage where an error may have occurred. This text area also provides the user with useful information such as the number of waypoints held in the GPX file. In the "Map" tab of the program a series of check boxes are displayed to show the user the data types (elevation, name…) contained after parsing that can be viewed on the map/graph. Every button in the program provides some visual feedback for the user in the form of a dialog or a visual response.

4. **Design dialog to yield closure.**
   Sequences of actions should be organized into groups with a beginning, middle, and end. The informative feedback at the completion of a group of actions gives the operators the satisfaction of accomplishment, a sense of relief, the signal to drop contingency plans and options from their minds, and an indication that the way is clear to prepare for the next group of actions.

The program consists of 3 stages: Loading a GPX file, Parsing the GPX file and viewing the results. At the end of every stage the next element is enabled on the program and the user is informed that they can move on. This linear set up allows the user to navigate through the system without any confusion.

5. **Offer simple error handling.**
   As much as possible, design the system so the user cannot make a serious error. If an error is made, the system should be able to detect the error and offer simple, comprehensible mechanisms for handling the error.

All error messages are written in a natural language format so that the user can understand and attempt to resolve them. System functionality is only expanded when the user has completed the necessary pre-requisite steps thus preventing the user from executing the map before the waypoints have been read in from the GPX File.

6. **Permit easy reversal of actions.**
   This feature relieves anxiety, since the user knows that errors can be undone; it thus encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data entry, or a complete group of actions.

All error messages in the program have been written in a way that helps the user easily locate the problem and correct it. As the program does not change the GPX file in any way there is no risk of damage or deletion. If a user accidentally loads the wrong GPX file, another file can be selected and reloaded which will replace the preceding file. All windows in the program have buttons that can return the user back to the main menu.

7. **Support internal locus of control.**
   Experienced operators strongly desire the sense that they are in charge of the system and that the system responds to their actions. Design the system to make users the initiators of actions rather than the responders.

Automation is only used when it is needed. The user has to load files manually and then decided whether to parse or not instead of an automatic load and parse function. The user can open and close the map or graph whenever they require. Every element of the system can be accessed separately.

8. **Reduce short-term memory load.**
   The limitation of human information processing in short-term memory requires that displays be kept simple, multiple page displays be consolidated, window-motion frequency be reduced, and sufficient training time be allotted for codes, mnemonics, and sequences of actions.

The map visualisation can display up to 8 dimensions of data at any one time allowing for greater understanding of data. The 3 sections of the system are separated in the interface by tabbed windows allowing easy navigation. When the map or the graph is opened, the rest of the system is occluded to prevent an excess of on screen data and therefore confusion for the user.

As can be seen from the above points, the system adheres to the majority of the Eight Points and can be classed as a good example of a visualisation. A disadvantage of this evaluation method is the possibility that one or more points may not be applicable to the system. In this case Point 2 which details the use of shortcuts is not relevant as the interface has been designed to be very lightweight visually with little need for using shortcuts.

Another form of evaluation can be gained from "5 cautionary tales of interface design" as defined by Foley & Wallace (1974). As can be seen from section III C, "User Interface" the system follows the defined cautions and can be seen as a good visualisation from that respect.

VI. Conclusion

The purpose of this project was to answer the research question: "is it possible to create a piece of software that can take GPX data as input and produce an output that is useful for the user?" The solution to this project aimed to provide a way to display captured route data from (widely used) GPS devices and provide a way to interact with the data including filtering and navigating. On top of this the system would contain a user-friendly interface that allowed quick navigation and understanding of the various elements of the system to encourage multiple use of the system.

A large variety of visualisation methods and theories exist from Shneiderman's TreeMap to Malone's theories of spatial memory. This project was centered on main research areas in the field of information visualisation as documented by Chen and Spence. For the system to be effective the most important areas of visualisation that had to be adhered to were *selection* and *representation* as the relevant raw data has to be extracted from GPX files and then formatted for visualisation. The final system implements these two requirements to a high standard which is shown in the 'offer informative feedback' section of the evaluation. The GPX parser extracts the required data quickly with a large route, containing around 1500

waypoints, taking only half a second to parse through. This provides no interruption for the user and prevents boredom and frustration, two effects of a bad user interface (Foley & Wallace, 1974).

By creating the visualisation in the form of a map underlay with the various waypoint information overlaid on top the program itself uses very little memory and hard drive space and is quick to load and navigate. This small resource footprint should allow the code to be adapted easily to run on mobile phones and PDAs. Perhaps the system could even be adapted to run in real-time together with a GPS device to allow instant viewing of the route a user has walked over. This combined with pre-loaded route timings could lead to a waypoint based exercise program.

The system has been created with the ability to load a live directory listing from a web application which allows GPX files to be downloaded and parsed directly through the web interface. This infrastructure has been put in place to allow simple future connectivity to other web applications so that new data about a route can be downloaded as soon as the file is submitted to a web server and thus allowing the easy sharing of GPX files.

The visual aspect of the system could be further enhanced by the use of three-dimensional maps. If the system could be connected to a subscription-based mapping service (or if a free service was created) that contains contours and land-deformation information then the user would be able to gain a greater understanding of the change in elevation over a route by viewing the actual change in topography over an area of land. This would improve the visualisation as another dimension of data could be added to the same screen as the GPX route. The disadvantage of adding a 3D element to the visualisation is the increase of system resources it would require to run including memory and graphics processing. This would decrease the target user base to those with dedicated graphics processors.

The results produced show that the system agreed with the proposed requirements, design and solution. The project is a mash-up incorporating existing software engineering visualisation theory and methods written from the early 1970s, the use of memory models defined in the 1950s and cutting-edge technologies such as web services and Global Positioning to create an original new system. By taking into account the aim of the project, the solution and the evaluated results the project can be deemed a success and the research question can be accepted.

REFERENCES

"Growth Trend for Conneted Navigation and RTTW Connected Navigation", *Connected Navigation, ABIresearch*, Oyster Bay, USA, Q4 2007, http://www.abiresearch.com/research/1002804-Connected_Navigation [20th October 2008].

Biderman, A. "The Playfair enigma: the development if the schematic representation of statistics", *Information Design Journal*, 6, 1, pp. 3-25, 1990.

Bryson, W. "Notes from a Small Island", *Black Swan*, London, 1998

Chen, C. "Preface," Information Visualisation: Beyond the Horizon, *Springer,* USA, pp 1-3, 2004.

"What is the Document Object Model?", Document Object Model, W3C, USA, 19th January 2005, http://www.w3.org/DOM/, [12th March 2009].

Foley, J.D. and V.L. Wallace, The art of graphic man-machine conversation, *Proceedings of the IEEE*, 62, 4, 1974.

Foster, D. “GPX For Developers”, *GPX: the GPS Exchange Format***,** Kirkland Drive, Stow, Massachusetts, USA, 24th November 2007, http://www.topografix.com/gpx.asp [11th June 2008].

Garland, K. “Mr Beck’s Underground Map: a history”, Harrow Weald, *Capital Transport Publishing*, 1994

Inselburg, A. “The Plane with Parallel Coordinates, *The Visual Computer*, 1, pp. 69-91, 1985

“JFreeChart API Documentation”, *JFreeChart*, Objects Refinery Limited, Hertfordshire, England, 6th November 2008, http://www.jfree.org/jfreechart/api.html, [12th November 2008].

Leland, N. “Dominant Light,” Exploring Color, *North Light Books*, Ohio, USA, pp 117, 1998.

Malone, T. "Towards a theory of intrinsically motivating instruction", *Cognitive Science, 4*, pp 333-369, 1981.

“About SAX”, *SAX*, David Megginson, The SAX Project, Canada, 05th May 2000, http://www.saxproject.org/about.html, [12th March 2009].

Miller, G. " The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information", *Psychological Review, 2*, pp 343-352, 1956.

Müller, F. “Traditional Techniques”, *Hypervariate Information Visualization*, Ludwig-Maximilians-Universität München, Munich, Germany, 2008

“About Us”, *OpenStreetMap.org*, Steve Coast, OpenStreetMap.org, USA, 16th December 2008, http://www.openstreetmap.org/, [16th December 2008].

Shneiderman. B. “Eight Golden Rules,” Information Visualisation, Harlow, England, pp 6-11, 2001.

Shneiderman, B. “Interactive Interface Issues,” Software Psychology, *Little,Brown & Co*, Toronto, Canada, pp 215-244, 1980.

Shneiderman, B. “Tree visualization with tree-maps: A 2-dimensional space filling approach,” *ACM Transactions on Graphics 11,* January 1992.

“Map-Of-The-Market”, *SmartMoney.com*, New York, USA, 18th June 2008, http://www.smartmoney.com/map-of-the-market/ [18th June 2008].

Spence, R. “Introduction,” Information Visualisation, *Addison-Wesley,* Harlow, England, pp 6-11, 2001.

“JXMapKit”, *SwingLabs***,** Sun Microsystems, USA, 15th October 2008, https://swingx-ws.dev.java.net/ [15th October 2008].

“GPX Schema Documentation”, Topografix**,** Stow, MA, USA, 19th July 2007, http://www.topografix.com/GPX/1/1/ [12th July 2008].

Tufte, E., *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, CT, 1983.

“Verizon Coverage Map”, *Verizon.com***,** New York, USA, 18th June 2008, http://www.rentcell.com/verizon-coverage-map.jpg [20th June 2008].

Wegman, E.J. Hyperdimensional Data Analysis Using Parallel Coordinates, *Journal of the American Statistical Association*, 85, 411, Theory and Methods, pp. 664-675, 1990.

Intelligent Web Service Composition

Student Name:

Supervisor Name:

Submitted as part of the degree of **Software Engineering** to the

Board of Examiners in the Department of Computer Science, Durham University.

Abstract – **Context:** Increasing availability of Web Services and the advent of the Semantic Web mean that cost-effective software solutions can be developed making use of these services as opposed to client side application-specific programs. This study aims to address the limitations of current Web Service technologies by providing an intelligent means of composing Web Services.

Aims: To observe and evaluate the advantages of using a composition application making use of Semantic Web Services, while taking into consideration any limitations which may be imposed, effecting the overall performance of the application, and detracting from users experience.

Method: Software was developed which allows for the discovery, composition and execution of Web Services from a remote server in order to meet user requirements.

Experiments were carried out in order to assess whether the number of operations over the network decreased the performance of the overall application to a point where usability was hindered. Comparisons were drawn with the same application performing entirely on a local machine.

Results: The outcome of the experiments show that while there is a slight decrease in performance when composing Web Services over a network, it is not drastic enough to be of concern. The usability of such a system requires a certain amount of adaptation on the part of the user however, the use of ontological search methods are beneficial over all.

Conclusions: The benefits which are offered by composing Web Services have distinct value. There is reason for concern over availability issues for Web Services, as the concept is quite new. As their popularity grows, so will the credibility of such an approach, and so will users' comfort with using such a system.

Keywords **–** Semantic Web, Web Services, Composability, Intelligent Composition Framework

I. Introduction

A. The Semantic Web Technologies

The concept of the Semantic Web can be seen as an "extension" of the existing Web. The main purpose of the Web as it currently exists originated from the desire to share information in Scientific groups. While the usage of the Web has developed into a much more complex system, it is still information driven. The goal of the Semantic Web is to provide more detailed information about the data which is on the Web, with a view to giving it a well defined meaning. (Shadbolt *et al.*2006)

These ideas originally developed in the business world, with the intention of creating “contract” software between companies which would reduce the amount of work load in inter-business communications. This therefore meant that services would be developed by the in-house development teams in the businesses concerned. In any situation which arises similar to this, there is a need to develop a set of standards which have to be enforced in order to ensure that all systems are compatible with one another. Owing to the fact that these applications would be Web based it was decided that most of the formats would be extensions of XML and therefore the concept of these services became entirely platform and language independent.

The proposed “Web Services” would reside on application servers and could then provide two possible forms of access, some dynamic web page acting as a portal for user access, or a

.

method for protocol access from other systems. By allowing protocol access, the value of Web Services became apparent, as a supporting technology to human interaction. The World Wide Web Consortium began development of a number of XML based languages in order to support the development of the Web Services, and move towards the goal of the Semantic Web. It is crucial that every Web Service has some means of having Semantic description applied to it, otherwise there will be no advantage over the current solutions available today.

The enabling technology for cross platform access to the services which is most widely used is the SOAP protocol (W3C, 2007). Emerging against a number of other possible technologies such as Java RMI this protocol was not only platform independent but efficient in terms of overheads, as the only information which needs to be provided for invocation of a service, is the address of the SOAP ports and the parameters which are expected. Both items which can be combined with the semantic description of the service.

The semantics of a Web Service are clearly important, and therefore a standard was produced to normalise these, this is in the form of Web Service Description Language (WSDL) (W3C, 2001). The description language specifies a base set of elements which must be present in all documents, and then extensions can be made to give further information. One of the most fundamental points is the binding protocol and the invocation address, as without this there would be no way to call the service. The document also contains information on the inputs and outputs of the service, and their types. This proves crucial when determining whether two services are composable or not.

The notion of ontology-based searching is an area of research which aims primarily to improve the effectiveness of keyword based searches through use of a domain model. By modeling a subclass hierarchy, extra search terms can be chosen in order to improve results. The technology which allows for this is the Ontology Web Language or OWL. (OWL, 2003). This is an XML extension which allows for the definition of classes and the relationships between them.

B. Motivation

As discussed above, Web Services are gaining momentum and the number of them which are available on the web is increasing all of the time. Due to the nature of web services and the fact that they have roots in the ideas of software reuse, it is a sensible and cost-effective solution to make use of the services rather than downloading individually pitched software on the users machine. The enabling technologies which are discussed above are all relatively new and therefore not as well established as some other protocols in Computer Science. This means that they have not been revised to the same level of quality as some other languages and technologies, and it is difficult to meet the demands of the Semantic Web vision with them. Examples of the desired tasks include ontological searching, service composition and general service quality requirements.

In an attempt to solve some of these problems, this project will make the best use of technology available along with some original methods in order to provide Semantic Web Service functionality. The project is primarily concerned with three areas. Service Discovery, Service Composition and Service Execution. By creating a system which allows for powerful smart searching of web resources, the user would have access to a wealth of services acting as building blocks to create new web applications without programming. When the technologies become refined enough this process will be easy enough for an end user to complete. This is outside of the scope of this project, which will primarily serve as a tool to show that in a developer context the technologies can work.

C. Project Deliverables

This project aims to address some of the most pressing issues in the domain of Semantic Web Service Composition at present. The objectives are as follows:

1) Minimum Objectives

1. Develop a basic GUI that allows access to web services and formation of a service flow.
2. Show semantic elements from a web service, including their I/O and data types (using sample services)
3. Demonstrate the meaning of composability for a sample set of services.
4. Evaluate the developed service composition, quality of service provided and general performance.

***2)* Intermediate Objectives**

1. The system will signal the user when an invalid composition is detected and advise solution.
2. Develop an ontology for a particular domain which could be used for searching suitable services when given a concept/context.
3. Program will be able to deal with composition of heterogeneous services in terms of their I/O data, with consideration given to any possible preconditions and effects.
4. To evaluate the compatibility of service composition.

3) Advanced Objectives

1. To develop a simple user requirement parser which can be used to intelligently find suitable services.
2. To develop an in-depth Input Output Precondition and Effect computing method for service composition.
3. To evaluate the computing method and system usability with regards to user requirements.

II.Related Work

A. Service Architecture

Service Orientated Computing is not a new concept created for Web Services, it has had uses in the past. Due to this research has been conducted into how best to use services and create them. One of the problems of this project is outlined by Budgen *et al.*(2004) stating the need for a style dictating the coding of services. The development of a framework Software as a Service (SaaS) attempts to solve this problem by meeting the definition of a Web Service in that the service should be *used* rather than *owned*, independent of language and stateless. This is in response to the idea of the Chung *et al.* (2003) definition of a service being "web-based applications composed of coarse-grained business functions accessed through the Internet" which is rather more limited in its scope. Developing frameworks which conform to such a specification is key to allowing the dynamic composition and execution of SaaS, termed by Budgen *et al.* (2004) to be "a demand-led paradigm".

The idea of SaaS also lends itself to the idea of software components, particularly in the domain of Web Services. A software component is define as an element which "conforms to a component model and can be independently deployed and composed without modification according to a composition standard". (Heineman & Councill

(2001) This is a fitting definition for the notion of Web Services in terms of this system, as they will need to subscribe to these qualities in order to allow for execution as part of a larger composition.

B. Web Service Composition

1) Logical Framework Compositions

The nature of Web Services is such that they revolve around the use of messages to communicate with clients, and ultimately one another where composition is concerned. In terms of Web Services there are four possible operation modes for the messages which they use; “one-way”, “notification”, “solicit-response” and “request-response”. (Medjahed *et al.* 2003) These determine the input and output sequence associated with the service. Inputs and Outputs form the backbone of composability frameworks in that they are the simplest check to make. If one service expects to receive four inputs but the composition only provides two then it cannot be valid. These message modes give an indication as to what the service requires, and from this a framework can be constructed. Medjahed *et al.*(2003) introduces the concept of a logical framework for processing service compositions. It is based on a layered model which can be seen in Figure 1.

-

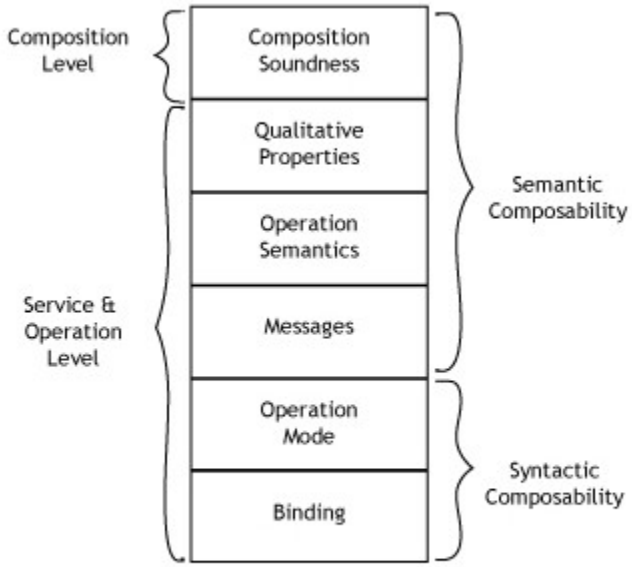


Figure 1: Layered Composability Model (Medjahed *et al.* 2003)

This model shows the each individual component which must be taken into consideration with regards to composing Web Services. However, by making use of SOAP the syntactic composability is dealt with, due to the cross platform and language nature the protocol. This turns all attention to the message composability. Message composability is the heart of Web Service composition, and after checking that the numbers match up, it is important to ascertain that the types do as well. In many existing frameworks, such as the one proposed by Medjahed *et al*. (2003) types are taken as strict rules. This is not always the case, and therefore this project will allow for a degree of flexibility in this area. By making use of the idea of “Composition Soundness” from Figure 1 the notion of a partially complete composition comes through. By defining this as a composition which has been subject to a type conversion, the user will be aware that results may have been altered, such as number

rounding. The only downside to this approach is the weight of programming which is required to compute the validity of a composition. The final system will use a heavily adapted version of the framework provided here in order to attempt to improve performance. This will include provisions for making sure that any Preconditions of the services have been taken into account.

2) Context-Based Compositions

As proposed by Medjahed & Yacine (2006) a context-based model could be used for composing Web Services. This would make use of the Semantics and ontology to its full extent. The motivating idea would be that Web Services become categorised and become party to context policies. These policies determine whether or not two services are suitable for composition. In terms of applying this scenario to automation, it would mean that many of the logic rules proposed would be replaced by these policies.

The framework which is proposed relies on different views of the Web Services in a strictly provider and consumer relationship. This is used in the formation of composition. The advantages outlined here are in terms of implied context. For example, data which is given to a system may have an implied context provided by the way in which it was submitted, (i.e. human interaction, output from another service or sensor) By utilising a degree of software intelligence, these contexts can be used without them having to be specified by the end user, ultimately making them more user friendly, and categorising them as Intelligence Amplifiers, supporting the actions of the user, by taking over tasks which are helpful to productivity.

This model is in opposition to the proposed logic based approach. By adopting a context-based approach there would be a need to create a number of "policy files" as detailed in above, and this would add overheads to the system, as well as providing another opportunity for deviation from standards. While the idea of being able to determine the composability of two services is purely in relation to an ontology is attractive, and would no doubt be faster in some cases than a logic based approach, it has not been proved as a reliable method as yet. When considering the notion of partial and total composability, the context-based framework does not place as much weight on the message levels of the composability model and therefore it may be harder to gain results in these areas.

While it would be possible to extend the context-based model to include some information about partial and total soundness of composition, this devalues the lightweight approach and would effectively be turning it into a more logic based approach. In terms of the proposed system there is some scope for making use of the context categorisation by ontology in relation to service discovery however. By taking into account that services can be grouped into "communities" then if one service is unsuitable for the requirements we would quickly be able to see that others in this community would not be compatible either. This would gain the benefits of faster searches but not compromise the ideas of partial and total composability.

C. Ontological smart searching

A relatively new area of research addresses such a situation as is raised with regards to combining the findings in Section II.B. By moving the context to the search engine, the composition framework can maintain effectiveness as a logical based approach. The work of Zhang *et al.*(2008) In finding a ontology-based model for geo-spatial information proves very useful in the domain of this project. In order to allow for effective searching, an ontology is used to define related terms. Then if a user searches for one term the paper defines a method for expanding this to include related ones. The model can be seen in Figure 2.

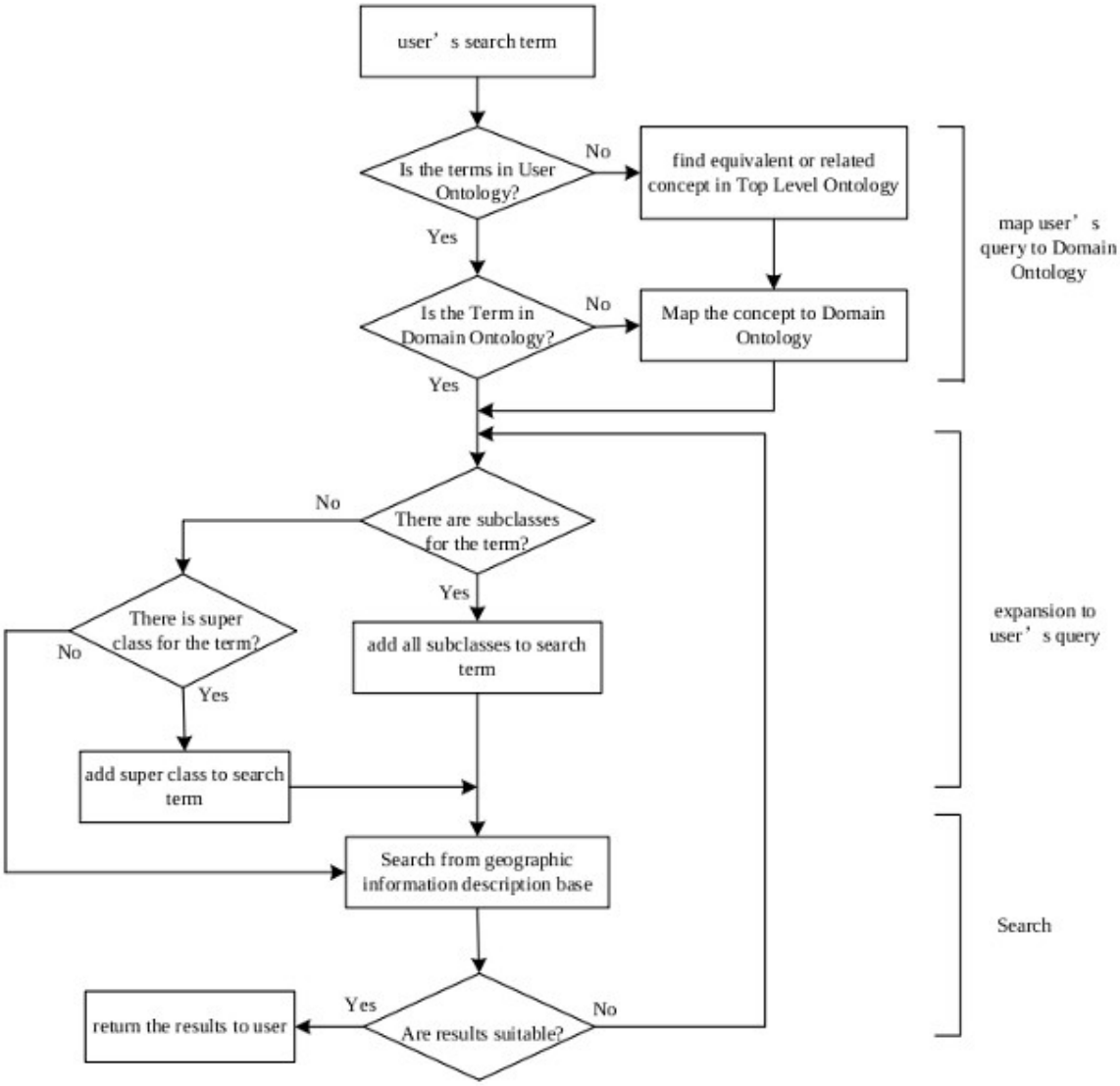


Figure 2: Ontology-based method for discovery and retrieval (Zhang *et al.* 2008)

The scope of the domain which this model hopes to address is far more complex than that of the project. Therefore it by adapting the algorithm to the needs of the program it is possible to see how a context-based search can yield more effective results than a traditional keyword based search approach. In applying this, the Quality of Service for the program will be improved, as the User Requirements will be more closely met. The work of Zhang *et al.* (2008) is inspired by Hochmair (2005) who, in the same domain, realised the need to develop an ontology in order to deal with the every increasing problem of "too-narrow or too large" search results based upon keywords. The nature of the data means that search terms are complex in this domain and this is the reason for the comparison of a user ontology with the domain one at the start. This won't be necessary in a more generalised application.

D. Summary of Related Work

Many of the problems which exist with the Semantic Web technologies have been researched in great depth, and in this chapter a selection of the best solutions have been discussed. All of them have limitations to some extent however, which this project aims to overcome. Building on the theories from the Service Architecture demand for dynamic manipulation of Web Services this will form the primary activity of the system. The technologies which are available do supply a great deal of functionality, however in many cases they require extra work to be performed in order for them to reach their potential. Taking the idea of Ontological searching for example. While the proof-of-concept research papers are good for the theory, the actual implementation of such an algorithm requires careful consideration in order to make sure that respect is given to all relationships within the domain, and non are overlooked – this would result in inaccurate results, undermining the purpose of its original application. By building on top of the logical framework provided by Medjahed(2006) the deliverables concerning service composition should be achievable from an early stage of the project. This will allow room for more advanced deliverables to be taken into consideration.

III. Solution

This chapter will detail the steps taken to achieve each of the deliverables which were set out at the start of this project. It will cover the relationships to the related work discussed in chapter II, and how this will contribute to the project. The chapter will be broken down to cover the components of the system and how they fit together within a layered approach, before discussing the major components on an individual basis, offering an explanation of how each meets a deliverable.

A. Development Process model

As is common with contemporary Software Engineering projects this system was designed and implemented using an iterative process model. This allows for changes to be made to the design and implementation throughout the whole course of the project in order to react to changes of circumstance or testing results. The system itself is built adopting the traditional layered model, defined by Sommerville (2006) as the data presentation layer, the application processing layer and the data management layer. This is particularly useful due to the distributed nature of the system which is demonstrated in Figure 3. The layers directly map onto each component, with the remote MySQL database server and Apache Tomcat Axis servers making up the data management layer, the client application composition and validation frameworks supply the application processing and the Graphical User Interface (GUI) is the data presentation layer.

1) Advantages of the Layered Model

By making use of the layered model for system design, other aspects of good Software Engineering are brought in. This includes code reuse. One of the key advantages to the Web Service technology is that it allows for the easy reuse of code in different circumstances. The client application also relies on this by making use of the JDBC driver for MySQL which is a third party component. Another advantage of making use of this layered model for the development is that it shares similar concepts with

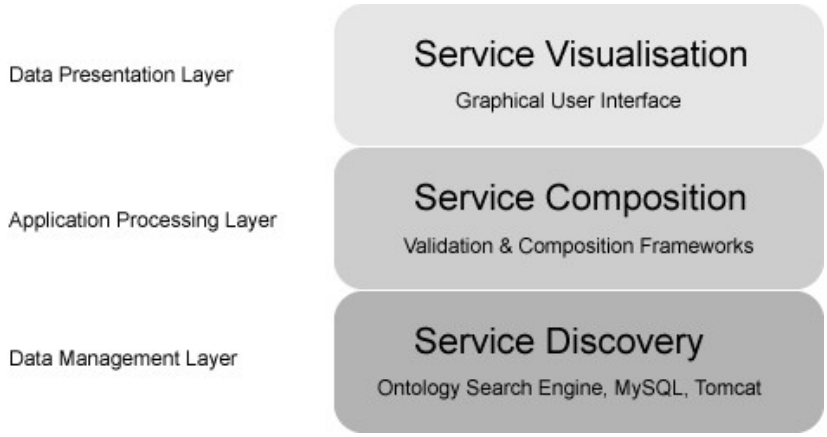


Figure 3: Traditional Layered Approach Applied to System

Component Based Software Engineering (CBSE). In CBSE there are clearly defined interfaces between all components which mean that they can be swapped without impact on the rest of the program. The system has been designed in this way, so that the layers are abstracted from one another. This not only helped throughout the implementation stage, as each component could be tested in isolation, it will also help from a maintenance point of view. This is due to the fact that the interfaces are strict and therefore program comprehension is much easier.

A clear advantage based in CBSE is that this development model lends itself perfectly to rapid prototyping. This is currently championed by the movement of AGILE programming, however in any case regular prototype releases are good for testing the system in its working environment and getting user feedback.

2)Possible Design Limitations

The fact that this software has a distributed architecture means that it is instantly open to reliability issues. In order to attempt to combat this more than one Tomcat server can be entered into the system, with all of the services becoming available. The idea is that if one server should fail, then at least the system would be usable to some degree, although the services offered may not be the same. In addition to this there is the risk of performance issues due to network latencies. This is a risk which has to be accepted in any project which uses network protocols, however since this project is directly concerned with the evaluation of performance for remote services it will play a large role in the results which are obtained.

B.Ontological "Smart" Searches

As discussed in Chapter II, the work of Zhang *et al.* (2008) is the basis of this projects ontological search techniques. The algorithm which is described there has been modified to some extent in order for it to meet the requirements of the system and to improve its performance as far as this system is concerned.

*1)*Differences to Zhang *et al.* (2008)

Where the model proposed by Zhang *et al* was designed to deal with a vast domain of complex terminology, the average user of a search engine will not need this scientific accuracy. Therefore there was no need to maintain a user domain to map onto the search domain, the system simply makes use of search domains. In order to make this

as generalised as possible the user is able to select the ontology which should be used with each search through the GUI.

One criticism of the model which was proposed by Zhang is that it relies on checks to occur at the end of its run time as to whether or not the results are suitable. Making use of the OWL property “disjointFrom” this system is able to recursively check super classes for disjointness with the current class. If non is found, then it is within the same concept and is safe to add, along with any sub classes. If a disjoint class is found, it is simply ignored. While this calls for a more formal structure to the OWL file than that of Zhang it also provides the possibility of gaining many more search terms. The ontology-based search in this system works purely as a keyword expander, taking the search input and returning a more appropriate set of terms where possible. Figure X shows the adapted version for this program.

*2)*The Search Algorithm

The application of an ontological search algorithm in order to enhance keywords may seem redundant when considered on a small scale. This is due to the fact that developers are used to using meta-data for objects, which specify a list of appropriate keywords. The problem with this is that with language differences and colloquialisms common in every language it is

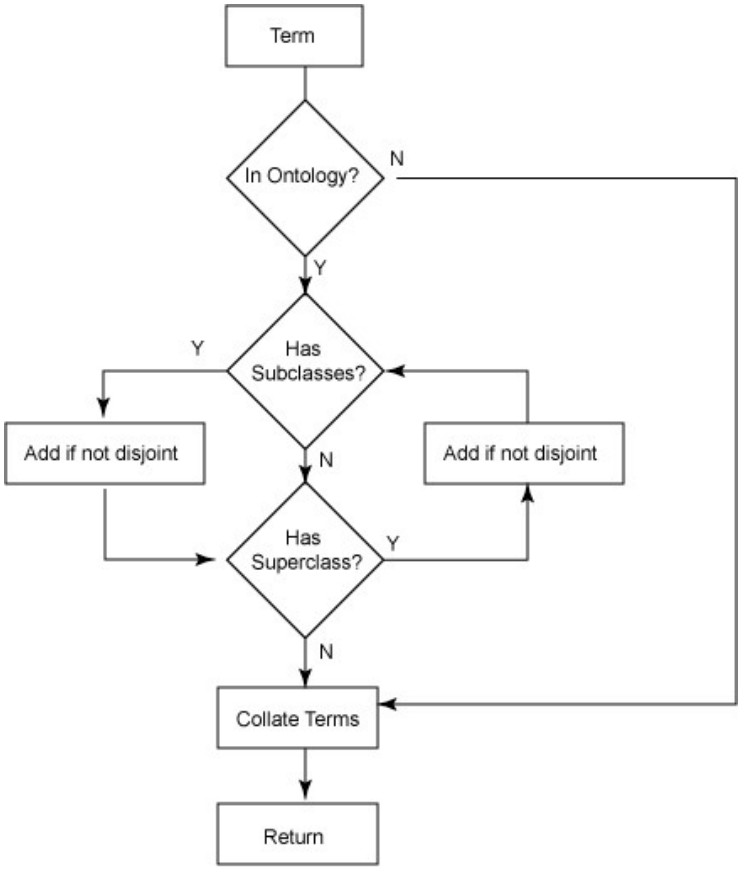


Figure 4: Adaptation of Zhang *et al.* (2008) algorithm

often hard to maintain an optimal set of keywords. One solution is to manually add everything that the developer can think of, but this is obviously not ideal. The design of this system allows the specification of one, or more, carefully chosen keywords to act as concepts.

The system can then interpret these and automatically generate the keywords to search for. The obvious advantage being if something needs to be changed, the ontology is changed and redistributed rather than every other service or object which needs updating.

*3)*Summary of Component

The ontological search satisfies both the intermediate deliverable directly relating to it, but also due to the nature of its implementation the advanced deliverable for a user requirement parser. The fact that the algorithm was implemented in such a way that the program "understands" the input requirements and modifies them in a positive manner this demonstrates parsing and giving feedback.

C.Service Composition

Two approaches to the problem of service composition were outlined in Chapter II both of which are feasible in their own right. The logical framework(Medjahed *et al.*(2003) based on a set of rules which determine the composability of a two services is a very formal method of solving the problem. It demonstrates qualities which the developer is used to using in software creation and the idea that everything is reduce-able to logic. An idea which is then challenged by a later paper by the same author relating to a context-based framework. (Medjahed & Yacine, 2006) the context-based framework would have its application in some domains, as discussed in Chapter II however it is not suitable for this project other than to serve as some inspiration for the way in which the ontology smart search was created. The system is therefore based upon a variation of the framework suggested by Medjahed *et al.* (2003).

*1)*Differences to Medjahed *et al.*(2003)

The difference between the implemented composition framework and that which is suggested by Medjahed is primarily a difference in technology. The services which are accessed by this system do so through an Apache Axis gateway on a Tomcat server. This means that all of the SOAP message creation is handled automatically upon making a service call. This extracts away the bottom two layers of Medjahed's composability model. This makes the implementation task much more straight forward.

Another key difference in these two approaches is the interpretation of "composition soundness". It is proposed by Medjahed as a notion of almost completeness, but passable. While this is largely its application in the developed system, it plays a more specific role. None of the composition frameworks which were not context based allowed for type conversions between the services. Most modern programming languages will not allow you to use a "double" in place of an "integer" and this can cause problems with dynamic services. Therefore the interpretation of partial soundness in this project is not incompleteness, but "of a lower accuracy". For example, entering a double – 5.2 as an integer would give you 5. This means whenever this number is used through the composition it won't give you an answer which is 100% accurate all of the time. However, this is one of the discussions in relation to the ideas of Quality of Service. The user is given a choice by the GUI as to whether this should be used or not. If it is then they have to be willing to take a lower quality result in exchange for the composition being allowed. Obviously there are still limitations even with this system as some data types are incompatible, but it takes a step in the direction of solving these issues.

*2)*Creating Composable Objects

When a service is discovered by the system, a JAVA object is created so that a database and a WSDL query are not needed every time the user wants to see a piece of information about it. This means that the response time is much lower. In order to pass on these benefits to the composition framework, the system creates wrapper objects called Composable Objects, which store all of the details of a composition as it is created. This means that the links from the outputs of the service, and where they are going are all known and quickly accessible.

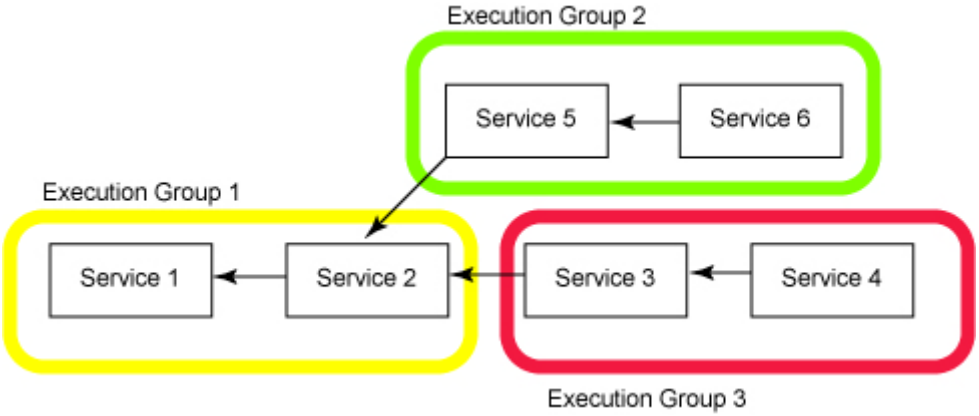


Figure 5 : Mock Service Execution Tree Demonstrating Precondition Computing

*3)*Summary of Component

The composability framework works based on logical rules as to whether the correct number of inputs and outputs are satisfied in a composition, and also the types. As discussed partial soundness is acceptable if two data types are compatible with each other. The composition framework is responsible for a large part of the deliverables of the project, from basic to advanced. It is a stable and intelligent component which integrates with and makes heavy use of the Composition Validation Framework.

D.Composition Validation Framework

The composition validation framework works along side the composition framework in the system. The purpose of this component is to perform the type checks and decided whether or not an acceptable composition has been made, utilising the rules outlined in the composition framework. In addition to this, it is this component which is concerned with checking the preconditions of services.

*1)*Service Preconditions

A precondition is something which must occur before the current service can execute. This is most commonly the execution of the service before it in the chain. Therefore it is crucial when deciding the validity of a composition that execution is possible in an order such that all services preconditions are met. The framework does this through the creation of execution groups which is an original concept not found in any of the existing framework research.

The basic idea behind an execution group is that in a simple flat linear composition where every service has one input and one output they form a chain. The only precondition is that the service is in the right place in the chain. This is what formed the basis of an execution group. The algorithm starts from the exit point of the composition and works in reverse, moving to the service which supplies it with input. In the event that two or more inputs are required, the current execution group is closed, and new ones are opened for each of the new services. These then continue until another fork occurs, when more groups are created. Each time a group is created, it is added as a precondition to the execution group which triggered its creation. Therefore the system deals with preconditions in groups to ensure that any chains which require parallel execution are successful in doing so. Figure 5 shows a mockup of a situation with three execution groups. In this situation groups 2 and 3 are preconditions of group 1.

*2)*Summary of Component

This component is crucial to the system as there would be no computing method for the Preconditions of services without it, and in turn their effects. In terms of mapping to the deliverables this component is under the intermediate category on a basic level, but the creation of execution groups suggests an in-depth method of computing preconditions, a requirement of the advanced objectives.

E.Service Execution

The service execution component plays two roles in the system. When a service is found through the search element, it is possible to test a service independently of any compositions. This is useful to allow the user to ascertain whether or not the functionality is what they are expecting. The real work of this component is done in the Composite Service Execution however. After a composition has been checked for validity and had the preconditions assigned, the execution component must make sure that the chain is executed in the correct order, and obey the composition frameworks decision on whether to attempt type conversions or not. Dynamic type casting is something which is not usually allowed in object orientated programming languages, and so a workaround was needed for this. Through use of the generic Object type in JAVA it was possible to gain the expected type from the WSDL file for the awaiting service, and use a set of logical rules to determine whether or not a sensible conversion could be made.

The service execution component is responsible for interfacing with WSDL files in order to determine the service endpoint and the way in which to administer a call via SOAP whilst passing the appropriate number of parameters. It must also keep track of the results coming out of completed execution groups, as they may be needed as a precondition to another execution group further on in the chain.

*1)*Summary of Component

While this component is not directly mapped to a deliverable, it would not be possible to measure the performance element of the system without executions taking place. The component will run services either locally or on a remote server depending on the address provided to it. Although this component is not responsible for determining the precondition/effects it is responsible for keeping track of them and the appropriate destinations throughout the execution phase.

F.Human Computer Interaction

In any software project it is important to consider the user at the design phase. There has been extensive research into the area of Human Computer Interaction, with the most prominent being provided by Molich & Neilson (1990). In Graphical User Interface(GUI) design, it is important to take into account trends which exist in other applications, such as the fact that the eye is automatically drawn to the top left hand corner of the screen first, before moving across and down in a diagonal move. Taking this into account, the interface of the system considers the three major components laid out in the order both which the eye moves, and in which they should be completed. Through the use of familiar symbols labeling the buttons and various confirmation messages, such as a plus symbol on buttons to add elements to the composition, there is a sense of familiarity with other pieces of software which the user may have used. It is crucial to ensure that the user can see when something is being processed in the software, and when the state of an element changes. Figure 6 shows the positioning of the composition status label in relation to the other composition elements of the GUI.

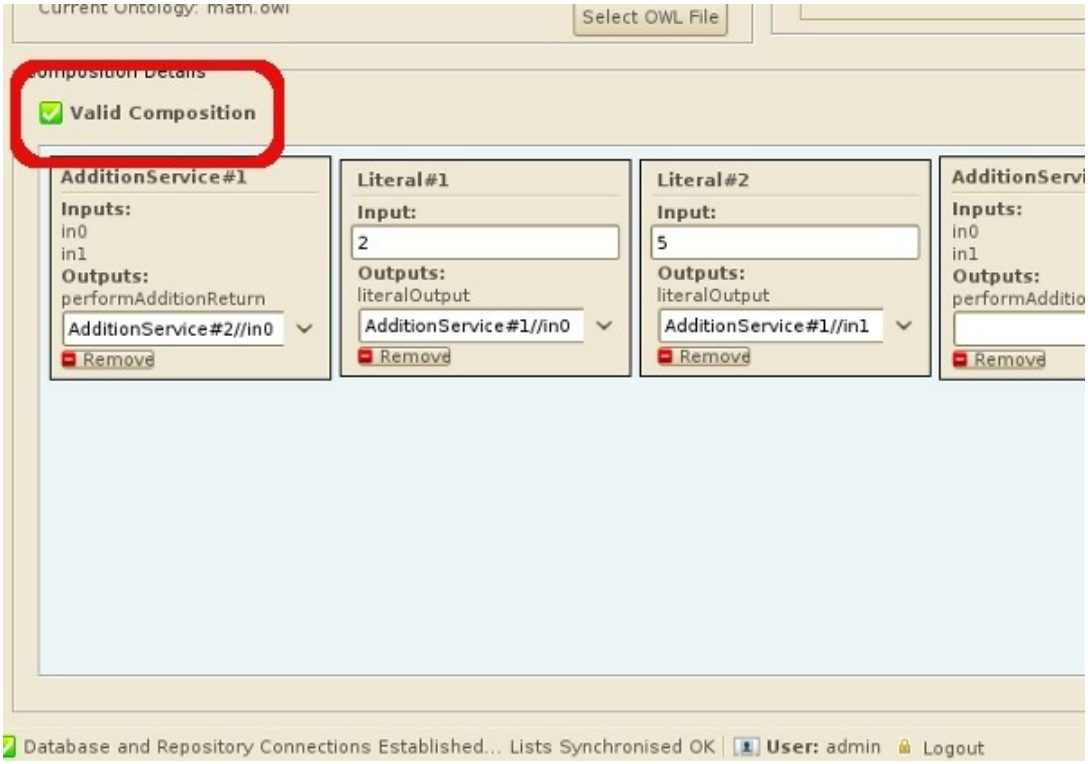


Figure 6. User Interface Design. Information label positioned in most effective place.

By placing this information in the top left hand corner of the composition panel, the user will automatically check it as a point of reference, as it instructs them on what to do next. In addition to this, the "execute" button is located after the validation button as this is the order in which the actions should be performed, and as one final measure the execute button does not become click-able until the user has created a valid composition. The reasoning behind this has its roots in the suggestion that good error measures are a key feature of any well designed GUI (Molich & Neilson, 1990). The definition of a "good" error measure should not be taken at face value as being clear and easily understandable. While this is important, it is far more important to make sure that you have an error message for every eventuality. While error prevention is a signal of good design, JAVA calls for the use of exception casting as a way of dealing with errors to prevent the program from exiting. Any system which contains such methods should have a corresponding GUI error message which does not give the full technical details to the end user, but gives a more generalised explanation and suggests something which may help.

Section III.E explains the methods of dealing with dynamic type casting in the system, a process which can cause errors if two incompatible types are found. Figure 7 depicts the error message which is presented when a number of type "double" is compared with an integer value. The error message explains that the problem is with data types, and offers a hint that allowing type conversions may help to solve the problem.

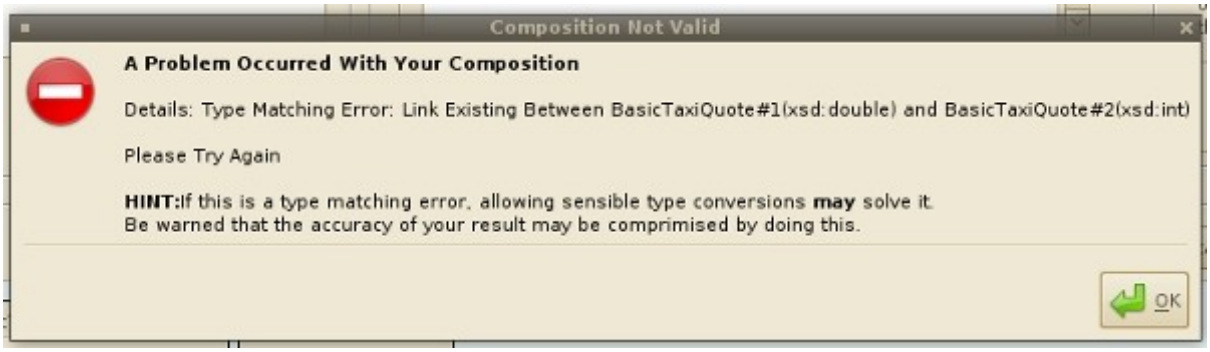


Figure 7. Incompatible type error with suggestion of fix.

IV. Results

A. Execution Time

In order to evaluate the advantages of using composite web services when compared with the same tasks running on the users local machine an experiment was needed. The hypothesis for this was purely concerned with the execution time, and was: "Compositions containing large numbers of services will ultimately take longer to run on remote machines than they do locally. This disadvantage will not be of a magnitude great enough to outweigh the benefits of the technology."

In order to collect the results the experiment was carried out as follows. Using one simple service as an example which remained the same for all executions, the service running locally was executed individually to begin with. Once the data had been collected for this, another four of the same service were brought into a composition. The validation of the composition was carried out separately and took an average of 0.5 milliseconds to complete. After running the experiment with five services in a composition, the same was tried with twenty locally running identical services, with a validation time of 9 milliseconds.

Once the data had been collected for the local experiment the same steps were repeated with the service running locally. Once again the identical service was used in order to keep the experiment as fair as possible. At the time of conducting the experiment there was no abnormal load on the network which could have affected access times to the remote machine. Table 1 shows the average execution times for each condition. The averages have been calculated from the results of running the execution twenty five times for each experiment.

TABLE 1. AVERAGE EXECUTION TIMES IN VARYING CONDITIONS.

Context	Average Execution Time (ms)	Minimum Execution Time(ms)	Maximum Execution Time(ms)
Single Local Service	17.85	10	42
5 Local Services	71.75	45	128
20 Local Services	198.30	133	277
Single Remote Service	351.4	56	5337
5 Remote Services	1674.55	306	5032
20 Remote Services	7482.85	1454	17730

In looking at the average execution time of both instances of single service execution, it is possible to see that the locally running code completed in much less time. The slight range in execution times in this instance will have been caused by the priority which the processor gives to JAVA in relation to any other running processes. Considering the averages of all of the local service executions in comparison with their maximum and minimum execution times shows that the amount of time taken to execute gently rises with the addition of more services. This is due to the overhead of executing preconditional services in the right order before supplying the end result. A similar trend would be expected in the remote services, with an allowance for network latency and data transfer rates. This was not strictly the case with the experiment. The execution of a single service had a much higher execution time than its local counterpart, although with a maximum runtime of 5337milliseconds, it is likely that the network was particularly unresponsive at the time of that execution.

The execution times of the remote services in general have a considerably wide range, peaking at 17730 milliseconds for the twenty service composition. While this demonstrates the unpredictability of network connections and data transfer speeds, it also shows that the running time is not of a magnitude such that it becomes infeasible to work with. Taking into consideration that when working with a composition comprising of 20 different services from remote servers, service objects are created upon their discovery and selection. This is a process which calculates all of the input and output requirements of the service and stores them in memory on a local machine, giving faster access time. Therefore the only time where there is a delay is when it comes to the actual execution and awaiting a result. In line with Internet connection speeds at present, even a wait of 7482.85 milliseconds, or 7.48285 seconds is not much longer than the average web page load time.

These results lead onto the hypothesis of a second experiment which was; “High execution times in relation to remote services are made up predominantly of network latency and data transfer rate resistance.” In order to test this the same service as the previous experiment was modified, and executed on the same machine. However this time rather than returning a result, the service actually returned the server side running time. This accounted for 3 milliseconds in the worst case and it failed to register a time on the JAVA “System.currentTimeMillis()” counter. The reason why this is lower even than the results in the previous experiment for one local service lies in the fact that this time the service was executed independently of the program and therefore did not rely on any of the communication protocols, it was purely the server side running time which was measured.

While this does not present an alarming drop in relation to the local single service run time, with a suggestion that seven milliseconds on average were used in communicating with the local service, it does prove interesting with regards to the remote single service. These results

show that only a maximum of three milliseconds was needed for an execution to run, which means that a much larger average of 349 milliseconds was required for the network communications between the program and the service. This grows to an average of 7422.85 milliseconds in overheads when considering the composition of twenty remote services.

B. Quality of Service

An important factor to consider for any service based software is the Quality of Service which is achieved through the application of this architecture. Combining the ontological smart search capabilities with the ability to perform sensible type conversions within the software results in a high level of service. The type conversions do lower the quality of the results in some cases, however this is the choice of the user through options within the GUI. If the user does select the option to allow for such conversions, the usability of the system is improved as it attempts to overcome one of the draw backs to having strict WSDL definitions for each message type.

C. Ontological smart searching

By allowing the user to not only select the ability to perform a smart searched based upon an ontology, but to also provide that ontology the search capabilities of the system have been greatly widened. In experimenting with the number of results which could be achieved without the need for specifying long lists of keyword meta-data for each service, the ontology search has proved to be incredibly powerful. With the previously discussed example of a mathematical term ontology, the search string can be expanded from the general “maths” to include the sub concepts of “addition”, “division”, “multiplication” and “subtraction”, including all of their respective subclasses containing alternative names. This results in a much more powerful search performance, due to the fact that most services performing “total” operations would be tagged as such rather than as maths. The ontology can then see that total is an addition operation and provide the user with other possible related services.

The searching experiment allowed for executing two different search queries one after the other, but both using different ontologies which caused no problems to the stability of the software or the quality of the results. The system is more than capable of dealing with a list of search terms, then delimiting these and processing each against the ontology individually without adding noticeable amounts to the running time of the search query. Although the algorithm supports scaling it is inevitable that some performance deterioration would be noticed should the number of classes within the ontology became larger than a hundred for example.

V. Evaluation

A. Service Execution

Any system which makes use of a distributed architecture over a network of any size is liable to fall prey to the nuances of such technology, and this project was regrettably no exception. While the experiments into the running time of the system proved that the time taken to execute large compositions of services on remote servers was not unreasonable it was still significantly larger than the time taken to perform similar tasks on the local machine. As the results showed when compared with the service running independently there is no doubt that the problem lies in the efficiency of the technological communication methods. While SOAP provides unrivaled advantages in being able to connect to and use a vast array of applications

regardless of their platform and programming language, it has large overheads in the message construction and parsing which is required as an extra layer on top of an already crowded OSI model for the Internet

The software produced in this project goes to some effort to lessen these effects however. As discussed in the running time experiment (Section IV.1) and leading to the hypothesis of the second, the system stores creates service objects as they are selected and explored. The disadvantage of this is that slightly more memory is used by the system, however the amount is negligible. The advantage of doing this is that the validation framework can execute with no delay, as all of the required information is stored on the local machine. This is preferable to wait times as the algorithm makes remote connections before creating the objects, which would theoretically take longer than the run times assessed in the experiment.

Additionally by storing a list of the services which are available on all current servers specified by the user, the keyword data for searching is available to the program much faster than having to extract it from the services each time. Although the database is remote to the program, the lead time on receiving results is far less than the alternative. Additionally, in delegating the query attention to a database server, any clients executing services will not suffer further overheads from users querying the database.

B.Ontological Search

The ontological based search component of the program, like the research area focused around it, is still in relative infancy. Some of the strongest work to date coming from Zhang *et al.*(2008) which was adapted to meet the needs of this system. The notion of concept based searching has obvious advantages, even through a relatively simple application, with time saved on the users behalf and more accurate, machine generated queries from a machine point of view. This work will provide the basis for future search engine concepts, as more and more meta-data is in effect wasted providing search engine optimisation. The greatest downside to using an ontology based search is the barrier to entry for many people. While it is conceivable that a provider could emerge offering fully functioning complete ontologies for some domain, it is not an easy concept for the larger user base to understand. Certainly creating ontologies which are in-depth and model complex hierarchies of concepts, can be associated with a certain degree of art in their creation. Should this idea take off as the new favoured way of performing searches, a more pure interpretation may evolve as opposed to the keyword expanding concept proposed and implemented by this system. The problem then lies in the future maintenance of the ontologies, ensuring that not only the terms, but the relationships which allow them to interact remain up to date and valid.

C.Usability

From the point of view of Human Computer Interaction (HCI) several features of the program were outlined in Section III.F as having being designed to meet theoretical ideas on the matter. The overall system feels somewhat familiar in its usage for the most part. Even the search engine which employs new technology integrates well into the system. The usability of the system is one area which is crucial should the system be pitched at a wide ranging user base. In its current state, the program would suit an audience of service developers who have some prior knowledge into the ways in which web services can be used. The reasoning behind this is although the vast majority of the system subscribes to the ideas of Molich & Neilsen (1990) regarding “Good Error Messages” there are some aspects of the system which may not be comprehended by a general user, even with the help of such messages.

One area of the usability which has been poorly implemented is the composition area. This is the area of the program which holds the services currently in the composition. As the

application stands at present this acts as list and nothing more, with the relationships defined using combo boxes on each service. This would be daunting to a non-expert user as they would have to select the correct input for each output to link to. The system does however, offer advice at the composition validation stage, should this have been done wrong. Due to the fact that a layered architecture model was employed from the start of development, this problem only occurs within the presentation layer, and could therefore be altered with little consequence to rest of the system.

The scalability of the system is promising. It can cope with searching through large numbers of service entries in the database with a reasonable running time. The experiments have proved that, although a little slowly, it will also deal with a composition of twenty remote services. In terms of the scalability of the system, it is not the stability where any issues lie. Due to the nature of executing composite services, the network overheads are practically unavoidable. A request must be made for each execution, any other solution would require some form of software acquisition to run the service locally, which negates the point of reusing Web Services.

In terms of security, a primitive role based security model is in place in the system. The purpose behind this is to protect the information in the database. The service browsing component will allow changes to be made to the description and keywords relating to a service, however an administrator must be logged in in order for this to be allowed. Any security concerns on a service level, for example banking, would be addressed in the implementation of the individual services for the most secure approach, this would therefore not require changes to be made to the system. As it stands, as a proof-of-concept essentially there is little need for further security than that which is already present.

Acknowledging the role of component reuse and iterative design processes proved crucial to the success of this implementation. Regular prototyping helped to spot problems as they arose and therefore deal with them on an individual basis. It also allowed for the system to be constructed and tested as separate modules. The vast majority of the deliverables which were set out for this project, with the only limitation being the absence of a method to store composite services to allow reuse. While this was not strictly set out as a deliverable it would make a nice addition to the project.

VI. Conclusion

The work which has been completed as part of this project explores several different areas of Semantic Web. From ontological search strategies to the logical composition frameworks the use of relatively recently developed enabling technologies form a large part of this project. Unfortunately this area of research is suffering from the same problems as some of its predecessors. In the absence of a strong, fast and reliable infrastructure upon which to build distributed systems, they will always perform second best to standalone applications, at least on paper. The danger here is that the advantages of such systems, approaches such as the one taken by this paper or the context-based approach seen in the work of Medjahed (2006) become quickly outweighed in the eyes of many due to speed alone.

The future of the Internet has its roots well and truly in the Semantic Web, and until more people begin to adopt the ideas it will always be second best in the eyes of many. The reality of the situation is that reusing Web Services in compositions as this program does, allows for the distribution of computing power, and resources. This cuts down on the cost of software development, and allows for a very convenient modular approach to software engineering and the maintenance associated with it. With the development of the enabling technologies into a more user friendly domain, and where possible with improved performance the work which has been carried out in this project could easily be applied to future projects. One possible work would be to combine the logical framework for composition, along with ontological searching, but extend this to a context-based framework for service description, such as the

work of Medjahed(2006). The resulting system would be far more powerful than the search and compose mechanism in this, however the groundwork is here to improve upon. As things stand currently, such a development would almost certainly end up once again as a proof-of-concept struggling to break through into a wider market. Although Web Services were quickly endorsed by business as a whole, much of interest has been lost, with companies such as Microsoft and IBM closing down UDDI servers, some of which only launched in 2004.

In conclusion to this project, future considerations of this nature must pay particular attention to the Non Functional Requirements of the system. This is where many of the problems which have occurred in this project originated, and is the reason why the progress of Semantic Web applications is restricted. The risk of reliability associated with depending upon web based resources is something which is becoming a part of many peoples every day computing life. This removes it as a potential problem, and while every effort should made to address reliability, new emphasis needs to be placed on speed.

REFERENCES

Budgen, D. Brereton, P. Turner, M. (2004) "Codifying a Service Architectural Style", *Proceedings of the 28th Annual International Computer Software and Applications Conference*,

Chung J-Y., Lin K-J. & Mathieu R.G (2003). Web Services Computing: Advancing Software Interoperability, *IEEE Computer*, **36**(10), pp35-37

Heineman G.T and Councill W.T. (2001). *Component-Based Software Engineering: Putting the Pieces Together*, Adison Wesley

Hochmair H, (2005). "Ontology matching for spatial data retrieval from Internet portals." In Rodriguez M A, Cruz I F, Egenhofer M J, and Levashkin S (eds): *Proceedings of the First International Geospatial Semantics Conference* Berlin, Springer Lecture Notes in Computer Science No 3799:166-82

Medjahed, B et al. (2003)"Composing Services on the Semantic Web" *The VLDB Journal*, DOI 10.1007/s00778-003-0101-5

Medjahed, B., Yacine, A.(2006) "Context-based matching for Web Service composition", *Distrib Parallel Databases* 21:5-37

Molich, R., & Neilsen, J. (1990). "Improving a human-computer dialogue: What designers know about traditional interface design". *Communications of the ACM*, 33(3), pp338-342.

Shadbolt, N. Hall, W. Berners-Lee, (2006) T. "The Semantic Web Revisited" *IEEE INTELLIGENT SYSTEMS* May/June 2006 pp.96-101.

Sommerville, I. (2006). *Software Engineering* (Vol.8).Addison Wesley.

W3C (2007) Simple Object Access Protocol (SOAP) http://www.w3.org/TR/soap .

W3C (2003) OWL Web Ontology Language overview.http://www.w3.org/TR/owl-features .

W3C (2001) Web Services Description Language (WSDL) http://www.w3.org/TR/wsdl .

Zhang, Q. Zhang, X. Li, D., (2008). "Ontology-Based Semantic Description Model For Discovery And Retrieval Of Geo-Spatial Information", *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol.**XXXVII**(B4):pp141-145.

Collaborative Multi-Touch Development

Student Name:

Supervisor Na

Submitted as part of the degree of Computer Science to the

Board of Examiners in the Department of Computer Science, Durham University.

Abstract –

Background: Multi-touch technology is rapidly becoming common place and offers many advantages over traditional human-computer interaction techniques. There is now the chance to discover new computer uses which utilise the more direct and intuitive input techniques offered by multi-touch.

Aims: To find if it is possible to implement a system that allows users on multiple computer interfaces (especially, but not specifically multi-touch interfaces) the ability to transfer materials between the interfaces in an intuitive manner using the user's knowledge of the physical locations of said interfaces.

Method: Concepts surrounding the aforementioned aims were developed and then implemented in two systems, a simple system comprising of simpler, more direct methods while as the other, more sophisticated system used more complex methods to facilitate implementations of the concepts that took advantage of the new opportunities afforded by multi-touch technology. These two systems were then compared and the concepts behind them evaluated through a controlled lab experiment.

Results: The concepts of the project were all successfully implemented between the two systems. Participants in the evaluation found it easier to get to grips with the methods used in the simple system but performed better (faster with a lower rate of encountering unexpected reactions from the system) using the sophisticate methods after getting used to the workings.

Conclusions: Users initially have trouble mapping from past experiences with existing input and material transfer methods to these new concepts but once the user grasps the new methods involved they are quite happy to use them in place of the current alternatives. With users taking longer to get to grips with the more sophisticated methods but using them more effectively the possibility of further developing these methods to be more accessible should be considered.

Keywords – Multi-touch, physical environment, collaboration, intuitive, transfers.

I. INTRODUCTION

This paper documents an investigation into the use of computer users' knowledge of a physical environment for facilitating collaboration between multiple interfaces. Multi-touch technology is rapidly becoming more and more common place (Han, 2005). This investigation focuses on discovering new methods for collaboration that could be used in conjunction with this technology for collaboration. Multi-touch technology has only recently started to be used widely in consumer products. One such product is the multi-touch table which is made use of as part of this investigation. This is a table with a multi-touch interface for a surface.

One of the main advantages of multi-touch is the facilitation of collaboration by allowing for more than one user to use a single interface simultaneously (Stewart et al., 1999). However as of yet, little research has considered the use of networked multiple multi-touch interfaces as a collaborative tool. The particular type of collaboration this investigation is concerned with is the fluid transmission of materials between multiple interfaces where the term materials refers to typical computer content such as documents, images, videos or any

other type of media. The motivating question behind this investigation asks “If a multi-touch table has information on its location and orientation in a physical environment relative to other tables, how does this affect user collaboration across all tables?” Also of great interest in this investigation is the use of multi-touch to make any developed methods and systems more intuitive. This section of the paper explains the scenario from where the aforementioned questions arose and states the objectives to be achieved from the investigation.

A. Background

As part of this investigation the possibility of using multiple networked multi-touch interfaces as a collaborative tool will be explored as multi-touch becomes a much more widely used technology it will be very likely that multiple interfaces will be used in scenarios where users wish to collaborate with each other in an intuitive manner with separate multi-touch devices (an example of this that is already happening is the increasing number of users of the Apple iPhone interacting with each other through use of the devices). The original idea for this investigation originated from the Technology Enhanced Learning Group at Durham University’s Computer Science Department. The group are currently researching the use of interactive multi-touch tables for education, specifically at a primary and secondary school level. A large number of the activities in the classroom require collaboration between students. Any applications developed for the tables as alternatives to these activities will therefore need to be able to communicate with each other between tables. In addition to these tables in the classroom vertical multi-touch displays are present for the teacher to display materials to the whole class with. An example of the applications being developed for these tables is the common classroom activity mysteries. For mysteries pupils working in groups are given a mystery and each group are given a different set of clues which the groups are then encouraged to share and trade in order to get enough information to solve the mystery (Leat & Higgins, 2002).

Currently applications for the tables are built using the group’s SynergySpace framework; a Java project built using a popular 3D game engine called JMonkeyEngine. The framework allows for a simple implementation of functions such as displaying materials (shapes, images, videos) and allowing the user to interact with these materials. The framework also offers simple networking features through the inclusion of existing Java packages such as those used for discovery and communication across a small network. A common requirement for the applications currently being considered and developed for the framework is the ability to transfer materials from one screen to another. An example of this comes from the mysteries task identified above. As part of this activity students are required to trade clues with each other in order to gain more information for solving the mystery. This paper discusses the problem of transferring materials between tables.

At the moment most material transfer techniques rely on surrogate representations of destinations and departure locations such as icons representing network locations such as the use of a shared network location represented through folder trees. This technique of material transfers is used in most of today’s widely used graphical operation systems such as Microsoft Windows, Mac OS X and most Linux distributions. These existing techniques require understanding of the representations and require additional knowledge, such as computer addresses and identifiers. In an educational environment these representations are undesirable as they add an extra level of complexity in using the system. Also school children may have problems using classical input techniques that rely on text due to their limited literacy. This investigation considers new input techniques, especially those now possible with multi-touch interfaces that can be easily performed and understood by younger users.

Tables are repeatedly moved around classrooms, in some places several times a day (Tibúrcio, 2005). This was important to consider when investigating techniques for configuring the tables as table orientation and location are required to be updated regularly. Also of importance is the presence of vertical multi-touch displays in the class room. When materials are on a multi-touch table they could possibly be rotated and arranged to face any direction on the display. On a horizontally flat surface this is not a problem as the material can be rotated for any user based on their positioning around the table (Kruger, 2004). However contents of a vertical screen (including the materials sent to it) are intended to be seen by many users. Materials should be aligned in such a way that all users can read the content from the same perspective. For example materials like text documents, videos and images can only be viewed correctly from one perspective so this investigation was required to find ways to accommodate for this.

B. Objectives

The objectives of this investigation centre on the aim of designing and evaluating possible solutions for the given problem, then evaluating the implementations of the chosen solutions. The investigation's objectives are expressed in three categories; minimum, intermediate and advanced. As some objectives called for enhancements or alternatives to the resulting methods from other objectives it was decided that two systems, a simple system and a more sophisticated would be implemented to cover more than one of the possible solutions. Table 1 lists these objectives and demonstrates how the objectives are applied to the two systems. Advanced objective 2 is later focused on in the evaluation due to its potential for original research.

TABLE 1. OBJECTIVES TO BE ADHERED TO THROUGHOUT THE COURSE OF THIS INVESTIGATION.

Minimum Objectives	1. Determine potential configurations of multi-touch interfaces for different purposes (E) 2. Investigate and implement a simple tool that can configure the possible table layouts manually (Si) 3. Investigate and implement methods that allow for materials to be transferred between the tables (Si)
Intermediate Objectives	1. Investigate and implement enhancements that can be made to the transfer methods (So) 2. Investigate and implement methods for retrieving materials from remote locations (B)
Advanced Objectives	Complete one or more of the following: 1. Investigate and implement a method for aligning materials on a vertical screen (So) 2. Investigate and implement a more intuitive method of setting up table layouts (So) 3. Investigate the possibilities of placing constraints on transfers of materials (So)
Key	*(Si) - objectives applied to the simple system, (So) - objectives applied to the sophisticated system, (B) - objectives applied to both systems, (E) - objectives applied elsewhere.*

II. RELATED WORK

This section looks at work carried out before on similar or related lines of work which have outcomes that are relevant to this investigation. This discussion first looks at how users tend to behave when asked to collaborate with computing devices in general and considers literature which examines specifically the use of a multi-touch interface to facilitate collaboration. Following this important developments and ideas concerning human-computer interaction (HCI) are considered, specifically for the use of multi-touch interfaces.

A. Collaboration with Computing Devices

Existing research concerning multi-touch technology focuses mainly on hardware and relatively little on collaboration. However there is extensive literature on computer based collaboration. This investigation focuses on drawing together these two disciplines. Some literature exists which focus on collaboration utilising single multi-touch interfaces. Though this investigation focuses on the use of multiple multi-touch interfaces for collaboration some points from this literature can be considered relevant. One such observation was that users will start to communicate indirectly using the interface (Rogers et al., 2004) implying that users do not solely rely on immediate face to face communication if there is an alternative available. This indicates that users on separate tables would rely more on the interfaces for communication rather than direct communication with the other system users. However the work this prediction originates from only concerns the results from a specific study and not the understanding behind user behaviour.

Peltonen at al. revealed how some users tend to take control of an interface device and dominate its usage even though fair usage is possible (Peltonen et al., 2008). However they have not considered the use of multiple interfaces for collaboration. This work lacked focus on other aspects in its study which could have been relevant to this investigation such as the consequences of this behaviour and methods to combat it. However the main focus on how users can become possessive over use of technology was a fact taken into consideration for the design of this investigation's systems. Peltonen et al. also dictates that giving users control over resources can cause problems such as where the software allows users to choose whether to allow materials to be sent elsewhere. Some users may choose to hoard materials when in other scenarios they would not (Peltonen et al., 2008).

Another important consideration concerns intuitive actions. Kosinen noted that users seem to expect the same user actions to be available in similar systems (Kosinen, 2008). For example, a flick motion which is used to move objects is commonly expected to be available. The idea of several interfaces utilising the knowledge of physical environment can be related to current research where mobile technology is used to make computational devices continuously aware of the locations of other devices (Kortuem et al., 2005).

Being an interactive classroom, collaborating students need to be able to share and find resources easily (Stahl et al., 2006). Kozma shows that the use of software for a collaborative exercise will encourage school children to discuss ideas and problems with each other rather than seeking help from a teacher (Kozma, 1999). This in conjunction with the findings of Scott et al. shows that collaborative software in an education environment can be greatly beneficial (Scott et al., 2003).

B. HCI Factors

Most current existing literature about multi-touch concerns either the actual mechanics of multi-touch technology or specific applications of multi-touch in other fields. The majority of other research concerns the frequent investigations into the development of user tracking techniques on a single interface (i.e. resolving to which user a particular input belongs to) and new gestures enabled by multi-touch (i.e. intuitive actions that users can easily and quickly pick up and perform). The concepts and methods developed from this investigation should be independent of hardware.

This investigation considers HCI factors in its design and implementation. Harper at al. state that with technology moving forward our way of designing it must move forward with it and the implication of this is that the methods used in designing interaction techniques for

systems using past technologies cannot be applied to new and future technologies without at least some adaptation (Harper et al., 2008). This applies to multi-touch as most HCI development techniques for classical computer systems assume one input at a time (i.e. a mouse location, mouse click or keyboard input). But with multi-touch this is no longer true as a user (or multiple users) could be pressing the screen in multiple places, i.e. simultaneous inputs. Also with multi-touch the input itself can be more expressive than simple clicking and typing, the design of touch based input systems need to consider the design of gestures more than the design of the graphical user interface (Nakagawa et al., 1999). Nakagawa's work on gestures is not applied to multi-touch environments but refers mainly to single touch technologies that were emerging at the time. However this work considers the understanding of a user's actions in relation to the ideas behind the technology used which makes it applicable to this investigation.

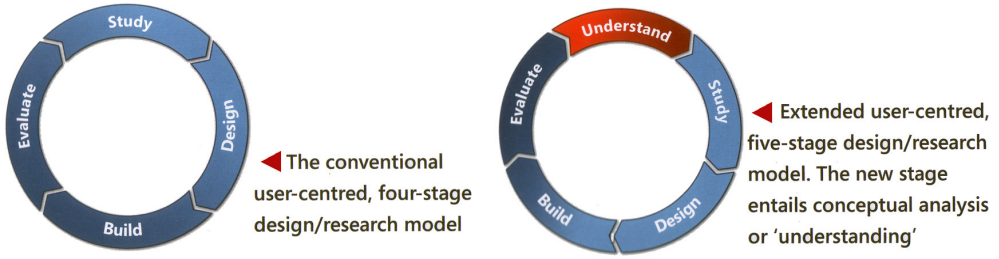


Figure 1. Comparison of existing design/research model with a proposed new model (Haper et al., 2008).

An investigation which concerns a new interactive technology should not be based fully on design methods and models intended for use with other input technologies. The investigation accommodated for this by adopting an enhanced design/research model which builds on the widely-used repeating sequence of study, design, build and evaluate by separating the act of understanding from the study stage (Harper et al., 2008) as shown in Figure 1. The understand phase is important to the aims of any investigation which aims to improve intuitive design. For this reason the evaluation of the investigation focuses not just on quantitative data but also qualitative so that the reason for participants' actions could be understood. Harper et al. made use of techniques such as 'thinking aloud' and questionnaire surveys of participants with open question allow for better understanding of users (Sharp et al., 2007). Sharp suggested that these techniques are mostly aimed at conventional interaction techniques but the information given on evaluation is general enough to be applied to this investigation.

Also an important possibility to consider when designing a user-orientated system is the ability to capture more information with a gesture than with traditional techniques which would require the same amount of effort from the user. So by using already collected information more efficiently (i.e. getting the system to put more work into calculating data rather than getting a user to input more) the resulting system can minimise the necessary amount of user input (Schraefel, 2006) which is greatly beneficial to the user's understanding and use of a system as it can make repetitive and time consuming tasks such as those that require large amounts of user input quicker and easier. Schraefel's experiment collected both in depth quantitative data and qualitative data and draws a number of conclusions from both: (conclusions go here). Schraefel's work demonstrates the benefits of collecting both quantitative and qualitative data in order to understand a user's view of any systems or their concepts developed as part of this investigation.

III. SOLUTION

This section details the design of the solutions to the given problem of transferring materials using a user's knowledge of a physical environment and describes the implementation of these solutions. The solution is broken down into four main subsections called solution parts: (i) the method of engaging the transmission of materials, (ii) the set up method of the tables, (iii) the method to retrieve materials from a remote table and (iv) the methods involved in aligning materials on a vertical display. The objectives call for some of these solution parts to be implemented in both a basic and more sophisticated versions to better test the concepts separate to the implementation. For this reason two systems were implemented. Simpler methods which are considered simpler for the user to understand are implemented in the simple system. The more complex methods, usually designed to make the most of opportunities offered by multi-touch technology are implemented in the sophisticated system.

The SynergySpace framework can be both run to respond to touch inputs on multi-touch interfaces (including simultaneous touches) but can also be run in a simulation mode on computers using conventional interfaces. This allows the software to respond to user clicks as if they were touches on a multi-touch interface. It is important to note that in this section of the paper that any reference to a user's touch will also refer to a user's click or any other alternative type of corresponding input method on non-touch interfaces running the systems. Also references to materials within the scope of the developed system refer to representations of shapes or pictures on the display.

A. Transferring Materials

The chosen methods utilise a designated area of a multi-touch interface named a transfer area. This is a gateway which relates to another table where correspondence is indicated through matching colours between the transfer area and the related table's border. When a material enters a transfer area it can then be sent to its corresponding target table where the material is placed in the transfer area on the target table which refers to the table from the material was dispatched from. This method requires little calculation and makes it obvious to the user where materials are travelling to and from. The simple system uses the basic drag and drop method familiar to most graphical systems. This corresponds to the minimum objective of having a method to transfer the materials. The adherence to the intermediate objective calling for a more intuitive method is demonstrated by the difference in the sophisticated implementation. It is important to note that SynergySpace can allow for materials to be flicked which is an intuitive action for touch-based interfaces (Kosinen, 2008). In the sophisticated system's implementation the ability to flick materials is used where users simply flick materials into the corresponding transfer area of the table they wish to send the material to. Transfer areas for this method can utilise smaller areas as materials only need to pass through the area rather than be placed in the area. This frees up more space in the table environment. The direction and speed of a transferred material is communicated as part of this method. This is so that on arrival the material can appear to continue along the same vector with the same momentum on relative to the physical environment.

B. Table Layout Set up

Table layout setup is arguably the focus of this investigation as it deals directly with applying a user's knowledge of a physical environment to inform the tables how they are set up. As previously mentioned the use of transfer areas is to be used for the transfer of materials. The

ultimate goal of this solution part is to set up these transfer areas. A simple way of setting up these transfer areas is through the user setting up the location of the transfer areas directly. By cycling through the tables in an arbitrary sequence a user is asked to set up transfer areas to other tables (which are also cycled through in the same sequence). The table currently awaiting input is highlighted with instruction. The table the next created transfer area will refer to is also highlighted though a change in its background colour. This is shown to the left of Figure 2 below (part 1). The user touches the border in two places (either simultaneously or one after the other) and the transfer area is created between the two locations. Transfer areas can exist either along one side of the table environment or across one corner of the environment. However since every table requires the user to set up transfer areas for all other tables a large amount of input is needed. This is improved upon for the table layout set up method used in the sophisticated system. The amount of input required for the methods selected is analysed in the evaluation section.

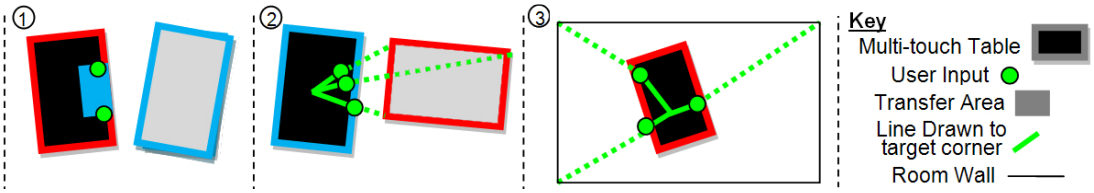


Figure 2. Three different possible methods for setting up transfer areas.

The possible solutions considered for the sophisticated method need to utilise intuitive set up methods in accordance with one of the advanced objectives. All the solutions considered were built on the idea of informing each table of its location. Then by using this information the system can set up transfer areas autonomously. Once locations are known tables can calculate distances to each other and their relative positions. Transfer areas can then be created where a line between two tables' centre points cross a table border. All location and rotation information referring to the physical environment is classified as world data whereas all location and orientation information in relation to the display environments are classified as local data. For the set up of the transfer areas the difference between the world and local data must be considered. As the transfer areas need to be created using local data the gradient of the line between the tables' centre points must be appropriately altered. This is done by using the real world angle of the tables change the world data to local data. The following considered methods use the idea of the user pointing towards reference points. Pointing is performed by touching the interface of a table. A line is drawn from the centre of the table to the position of the touch so that if the vector of the line is followed from the centre the reference point should be found.

In the centre of Figure 2 (part 2) a method which uses the location of other tables to calculate its own location is shown. The user is required to use only one table for this set up method. The other tables are highlighted in sequence and the user points to 3 corners of each highlighted table. Using this information the table can figure out its location. However all locations and orientations are relative to the location of the table the user is performing the set up from. While the user is not required to travel between tables there are issues with the method's calculation of positions which make it undesirable. This is because the angles of the 3 lines to a table's corners can refer to the table being in several different configurations (i.e. one set of inputs could refer to tables being in several different positions with different orientations).

An alternative is to use fixed reference points in the actual physical environment. This has the benefit that users may find it easier to utilise and understand fixed physical locations.

Another benefit to using the tables' shared environment for reference points is that the world data is absolute and can be easily checked for correspondence with data calculated by the system. Methods that use the real world environment for reference points can have their location set up independently so tables can join at any time. This means that tables do not have to be set up at the same time and can also be setup in parallel. The first considered method refers to three reference points positioned on the corners of a rectangle encompassing the tables such as a room's corners. This is shown to the right of Figure 2 (part 3). However this method can lead to ambiguity in a table's position and without further inputs can not be specific about its current location (i.e. the inputs for several table locations and orientations are the same). This can be improved upon though to guarantee a correct location and to also collect a table's orientation. The setup is performed in two stages which can be implemented in any order. The first stage asks users to point to two reference points either simultaneously or one after the other. The other stage requires users to point in a direction parallel to the line between the two reference points. This direction is called the environment's north and is the same direction for all tables. While a difficult concept to explain to user and once understood it allows for tables to be set up quickly without dependence on other tables. For these reasons it is the selected table layout set up method for the sophisticated system.

This method is one of the main focuses of the investigation as it bares the most relevance to the research question. For this reason it was important that when it came to the evaluation of the system to focus on the accuracy of the results from the user inputs. This was to determine if the method was usable or if it is too dependant on user judgement. The following explains the workings of the method in detail. In these calculations only the x and y co-ordinates of user inputs and the distance between the reference points (collected from a flat file) are initially known. All angles shown are in degrees for simplicity but are implemented in radians for accuracy.

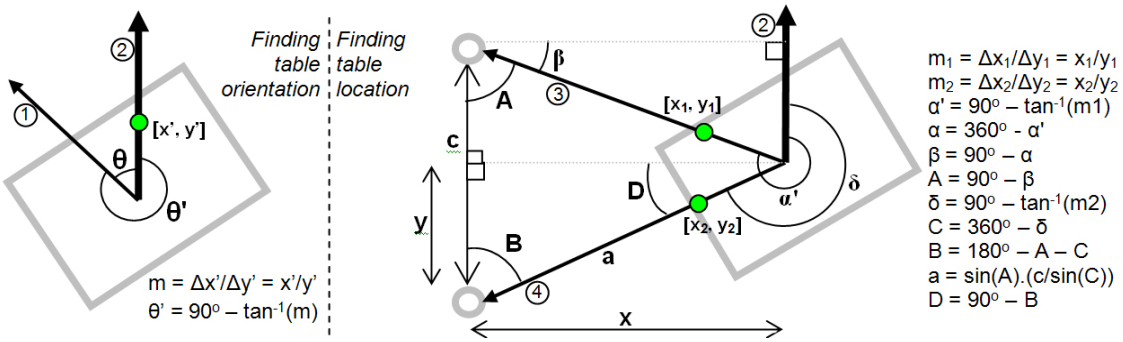


Figure 3. Measurements involved in the calculation of the table location and orientation in the sophisticated table set up method where only the local co-ordinates of the user inputs are initially known.

On both sides of Figure 3, line 2 points to the physical environment's world north. Eq. (1) below represents the final calculation to find a table's orientation and refers to the values on the left hand side of Figure 3. In this figure m refers to the gradient of line 2. The co-ordinate of the user's input local to the y axis travelling along line 1 (the line pointing to the table's local north) is represented as [x', y'] with the centre of the table being [0, 0]. Finally θ refers to the table's orientation in the world environment.

$$\theta = 360 - \theta' \quad (1)$$

Eq. (2) below represents the calculations used to find a table's location and refers to the values on the right hand side of Figure 3. In this part of the diagram m_1 refers to the gradient of line 3 (the line pointing from the centre point towards the most northerly reference point) and m_2 refers to the gradient of line 4 (the line pointing from the centre point towards the

least northerly reference point). The distance between the reference points is represented by c. and $[x_1, y_1]$ and $[x_2, y_2]$ are the co-ordinates of the user inputs. The co-ordinate of the table's centre point in the world environment is represented by [x, y].

$$[x, y] = [\sin(B).(a/\sin(90)),\ \sin(D).(a/\sin(90))] \quad (2)$$

C. Retrieving Material

The objectives call for a method of retrieving materials from a remote table. This problem breaks down into two smaller problems. The first problem is that of finding what materials are on which remote table. The following problem is that of getting those materials from the remote table to the user's current table. There is no call for a simple and an enhanced implementation for this solution part so there is only one developed method. However changes can be made between the two implementations to demonstrate a possible method for placing constraints on material transfers. This is called for by one of the advanced objectives and is explained later in this sub-section.

The chosen solution requires users to select which table they wish to view the materials from by touching the transfer area which relates to the table they wish to see content from. This action causes a box to be shown in the table environment in which the contents from the target table are shown in. This is the chosen technique as it makes the display of materials much clearer to the user. Users may need to do this for multiple tables if searching for a specific material. The display of the materials in the box could be done in two ways. The first possible method is to configure the materials to have the same configuration as the corresponding materials on the target table. However this suffers the problem of scaling the materials down with relative distances between them would make it difficult to find and select specific materials. Instead the chosen method is to place materials into a grid inside the box. The grid size is based on the box size. The number of columns of the box is derived from the ceiling of the square root of the number of materials. The number of rows is based on the floor of the square root of the number of materials. This ensures there are enough cells for all the materials to be displayed in with the number of empty cells being minimised. All materials are orientated the same way and any materials larger than their cell are scaled down. This shows the materials in a clear manner that users can easily employ for finding specific materials.

For the simple system users simply touch the material they want transferred and this is removed from the target table and placed on the currently used table. This is straight forwarded but there is a problem that the user on the target table may not have wanted to allow that material to be taken. This implies the need for a constraint on the transfer as called for in the advanced objectives. This is accommodated by the implementation in the sophisticated system. When a user touches the material they wish to retrieve instead of just taking it the material on the target table become highlighted. The highlighting of a material is where the material pulses and gains a border of the same colour as the table a request originates from. Also on the target table the transfer area relating to the table making the request is also highlighted. Users on the table can then see which material is being requested and can see which table wants it. The users on this table can then make the decision whether to transfer the material to them using the flick transfer method or to keep the material. The request times out after a few seconds and the highlighting of the requested material stops. . This makes the method more of a technique for requesting materials rather than directly retrieving materials which fulfils the advanced objective for implementing some form of constraint on the transfers in the system.

D. Vertical Displays

In accordance with the advanced objective calling for displaying materials correctly on a vertical display a method of aligning materials needed to be implemented. In the simple system no method for alignment is needed as the materials cannot be rotated. The sophisticated system however does allow for the rotation (and scaling) of materials. This means that for the sophisticated system no assumptions can be made about the alignment of the materials. Any solution to the problem will involve setting all material rotations to match the orientation of the display which is most likely to 0^o for a classically aligned display. However just setting all materials to have a 0^o orientation is not enough as then we could have a problem with materials overlapping with each other or the display edges.

The chosen solution is derived from the previously mentioned method for retrieving or requesting materials where materials were aligned in the showcase. By creating a grid with its number of rows and columns based on the number of materials currently on the table and its dimensions based on the display size the materials could be placed into cells of the grid after being rotated to the correct alignment. Any materials larger than their cell are scaled down to fit. This is the chosen method as it is always guaranteed to align the materials correctly in such a way that there are no overlaps. The method will always perform this configuration in linear time (based on the number of materials) and although the original configuration of the materials will be lost it is worth this trade-off for the running time and guarantee of a correct orientation. The implementation of the chosen method in the sophisticated activates from the user touching the screen borders and will align materials as if the border touched was the bottom of the environment. So on any display the user can tap the border on any side of the display which they perceive as the bottom relative to them to align the materials correctly. This gives the possibility to align the materials in four directions at any time and can also be used on the tables as a method of sorting the materials quickly if current configuration of the materials is deemed too untidy by the users.

E. Networking

Though not a part of the previously mentioned solution parts networking is vital to the implementation of the system. The physical type of the network doesn't matter but the methods and protocols used for the communication of messages between the tables are important. The SynergySpace framework at the time of development of the system utilised several Java networking libraries which facilitated the design of the networking and had several classes in place to implement these. The implementation of the systems built upon and extended these classes for full use in the system. All the networking transactions were run in a thread separate from the code which generated the display and handled the user inputs. Due to this the design needed to be very aware of concurrency. For this reason all messages being sent or received were stored buffers between the update loop in the networking thread and code utilising the SynergySpace framework's update loop. The tables are able to discover other tables using the same application through the use of a Java discovery service library. The service names are derived from the application running so tables can only discover and connect to tables running the same application.

Initially the design of the system called for a peer-to-peer network as all tables would have the same functions and influence over each other. However the protocol most suited to this structure is User Datagram Protocol (UDP) which is undesirable for this investigation as data on large materials needs to be transmitted without any loss. Transmission Control Protocol (TCP) was decided upon to be the networking protocol used due to its low loss of

data in comparison to the alternate method of UDP. However using a peer-to-peer network with this protocol does cause problems. One such problem is the broadcasting of messages which the systems need to be capable of doing. This is very demanding both in time and network resources with TCP so it was decided to use a server-client network structure. This was implemented by setting the first table to run the application becomes the server. So a table which upon searching the network for similar services discovers none within ten seconds designates itself the server. This server runs in the same way as any of the clients except the running of the server thread in the background so to the user this makes no difference. All clients upon discovering the server connect to it with a TCP connection. Now any message sent goes to the server which then sends it to the table it was meant for (including itself). Also upon receiving a message the server can send it to all tables in quick succession thus simulating a broadcast. This is less demanding to network resources than a peer-to-peer structure as the only connections needed are those between the server and the clients. Messages are structured and tokenized the same way so they can easily be parsed to find their origin and intended destination locations.

The next consideration for the networking is the question of when to send the material information. Sending it when needed (i.e. when a material is transmitted from one table to the other or requested remotely) causes a delay in the transfer. This is an unwanted trait as it detracts from the responsiveness of the system that is desirable for a system built around intuitiveness. So it was decided to send all present materials as soon as a table is set up. The materials are broadcast to all other tables and stored on these tables but not displayed. While using up more memory on each table it increases the responsiveness of the system. When a material is transferred by a user only the material's identity and current information is sent. Current information refers to data such as the material's location, rotation, scaling and momentum. The material information sent depends on the material transfer method being used. The receiving table can then recall the material from memory and appropriately apply the current information to it. The question of how to actually send the data on the material is the next problem to be considered. Initially in development the use of a Java library called XStream was considered. XStream allows Java objects to be parsed to and from xml representations. This would have allowed for the materials to be parsed to xml, transmitted as text then reconstructed when received by the target table. However the XStream parser took into account the information from all the classes the material objects extended and inherited from so when parsed the xml the resulting string were too large. These large strings on transmission caused networking issues due to monopolising the network connections and also took up far too much memory. This was undesirable, especially as so much of the information parsed and stored was not relevant so a specific serialisation method was written for material objects. This parser creates a string of data from an input material which only relates to the appearance of the object so no redundant data is duplicated. Using this information the parser can create a new material object from the correctly formatted and relatively short string of data. The formatting of this string is similar to the tokenised messages used in the systems' networking so transmitting the data is straightforward.

F. Implementation

Implementation of the systems as mentioned before was performed using the SynergySpace framework. Using this framework objects can be generated in a 2D environment which can then be assigned listeners. These listeners will execute given code when they receive a specific input such as a user touching an object for a specific amount of time at one or more locations on the object. Some of the listeners given by the framework involving moving,

rotating and scaling the objects they are attached to.

Throughout development backups of the code were made whenever significant milestones were reached on hard and flash drives separate from the drives where development was taking place. Also a repository was used so at the end of each development session the updated code was committed. A small level implementation issue encountered during development concerned the rendering of shapes which were created from extending a SynergySpace class. The current method for displaying shapes was to draw the shape on a 2D canvas similar to that used in the Java2D library and then use the canvas as a texture for the object. However to display images the shape class was used but the canvas texture was swapped for a buffered image object. This caused problems in the highlighting of any image with a picture. This was because highlighting of the shapes is executed by drawing coloured elements on top of the 2D canvas which was not possible with a buffered image instead of a 2D canvas. Also the process to serialise the data of an image shape for transmission involved separate code to the method used to serialise all other shapes. To solve this the method of displaying image was changed so instead of swapping the texture the image was drawn onto the 2D canvas using methods similar to the ones used for drawing the other shapes. This allowed the object to use the same serialisation and highlighting methods which had been used for displaying all other shapes.

The first implementation of the investigation was a tool for representing the possible layout of the multi-touch tables. This was both a test to see what was possible with the SynergySpace framework and an adherence to the first two minimum objectives which called for a study and demonstration of the possible layouts of the tables. The simulation works with multi-touch and traditional interfaces and displays tables as rectangles on screen. The tables are coloured using a colour allocation method developed to facilitate parts of this investigation. The colour allocation method is reused for the colour allocation in both the simple and sophisticated system. This simulation allows users to add as many tables as they like and can move the tables anywhere within the display. Users can also rotate the tables a full 360^{o}. This allows for all possible configurations of the tables to be viewed by the user and demonstrates how flexible the set up methods of the systems need to be to accommodate for these possible layouts. The simulation tool can detect when tables overlap as this is not possible in the physical environment so should not be accepted in the simulation.

IV. RESULTS

This section details the evaluation carried out of the two systems developed. The first sub-section concerns the actual set up and design of the experiment used for the evaluation. The second sub-section details results from the experiment.

A. Experimental Design

To evaluate both the implementation of and concepts behind the two developed systems a controlled laboratory experiment was carried out. It was decided that at least 20 participants should be involved in the experiment. From using quantitative and qualitative data the implemented systems and concepts could be evaluated to see if the objectives of the investigation have been completed. The sampling frame of the participants was comprised of staff and students at Durham University. Most, but not all of the participants were members of the university's Department of Computer Science. Participants were informed of the study either through social networking sites, e-mail or word of mouth and were sampled using non-probabilistic convenience sampling. The experiment was carried out by running the

developed systems on four tablet interface computers. Though not multi-touch these devices used a single-touch based interface. This meant that while the true benefits of multi-touch would not be evaluated the flexibility of the design and concepts would be tested. The testing of the concepts is of great importance to the objectives of this investigation as they relate directly to the research question.

Before the sessions took place the tablets were placed in positions likely to elicit the largest difference in user reactions and inputs. This required each tablet to have a widely differing combination of location and rotation. Before the sessions begun a dry run was performed to check the equipment and software. For this dry run no results of the tasks performed were recorded by any means so as not to cause a threat to the final results' validity. As a result of this dry run several changes were made such as connecting the tablet computers to a wireless router separate from the university's wireless to minimise the possible threat of network issues. Changes were also made to the sophisticated system's method of creating transfer areas. This was because previously the size of the transfer area (the amount of the table border covered by the transfer area) would depend on the distance of the target table from the current table. The further the tables were apart the smaller the transfer area. However the distances involved caused some transfer areas to be too small so it was decided to give all transfer areas a fixed size in the experiment. Precise measurements of the tablets' locations and orientations were made and recorded. This was for use later in comparing the given locations and orientations resulting from the participant inputs to evaluate the participants' accuracies.

Participants were invited to spend twenty minutes in a lab in Durham University's Computer Science Department with the tablet computers. Here they were firstly asked to complete a consent form and pre-session questionnaire. The purpose of this questionnaire was to collect information on the user's experience with technologies relevant to the investigation (traditional computing devices, multi-touch mobile devices, etc.). This information was collected for qualitative data. Qualitative data was considered important for collection as the comments could aid the understanding of user actions. This understanding would be useful for further iterations of the design/research cycle (Harper et al., 2008). Half the participants were asked to use the simple system and the other half were asked to use the sophisticated system. Participants were not told which system they were using to avoid bias. Each session was recorded from two cameras and a microphone so timings could be verified and more in-depth observation of user actions could be made at a later date. Users were made aware of the cameras and permission for being recorded was asked for as part of the consent form before the session begun. Users were also encouraged at this point to 'think aloud' so their comments on their opinions and actions could be recorded by the surveillance equipment to aide later observation. A brief demonstration and explanation of the system being used was given to each participant at the start of each session. Throughout each session of the experiment a demonstrator was on hand throughout the entire session to answer questions, take notes and offer advice to the participant where necessary.

Participants were first asked to complete the set up stage of the system they were using. The times taken overall and per table to complete this task were recorded for quantitative data. In the sophisticated system the participant's input parameters (location of inputs in the local environment) and resulting positioning information calculated was logged by the software. After set up users were then given some time to get used to the functions offered by the system used such as material transfers and requests. Understanding of the user was tested through a series of short tasks such as the demonstrator making requests to tables and asking the user to respond appropriately. In each session users were asked to transfer all materials to one screen. As the number of materials generated by the systems is random instead of

recording the overall time for the transferral of materials the average time for a single material to be transferred by a user was recorded.

After performing the given tasks users were asked for any final comments by the demonstrator. Any comments made were made these were recorded on the surveillance equipment and noted by the demonstrator. Participants as the last task in the session were then asked to complete a post-session questionnaire. This questionnaire consisted of a series of closed questions for user ratings of the techniques used with optional open questions so participants could make any extra comments on the system if they desired. If there was more time available at this point in the session then the participant would be asked to perform the same tasks again with the alternative system (i.e. the system they had not used previously). This crossover was purely for observation and qualitative data based on comparing the systems. No quantitative data was recorded from these parts of the sessions as it would be unfair to use the measured data from users with experience of either of the systems alongside data from users who were using the systems for the first time. To ensure privacy of all details participants were assigned randomly generated identification codes and only had their names recorded on the consent forms. All video and audio recordings were destroyed once all the necessary results were noted and no copies of any of the user forms or recordings were made. All questionnaires were shuffled to make any analysis of the results blinded so no connections or trends could be identified outside of those revealed by the recorded data.

Before the running of these experiments a hypothesis was created. This hypothesis stated that though participants may take longer to gain familiarity with the sophisticated system their use of the system's methods will be quicker and less error prone than their use of the simple system methods. The hypothesis therefore predicted that participants would prefer to use the sophisticated system after getting used to its workings. This implies the null-hypothesis which states that participants will prefer the simpler methods found in the simple system or that participants will prefer traditional methods for the transferring of materials.

B. Experiment Results

From the information collected during the experiments data was organised to assess the main focus of this experiment which is the participants' use of the two systems. The data collected also allowed for an evaluation of the sophisticated table set up method which is of great importance to this investigation as its evaluation is relevant to the research question. The term first reference point refers to the reference point which is the most northern in terms of the set up method. The calculated distance between the table and the first reference point was compared to the measured corresponding physical distance. The table's calculated orientation was compared with its real world equivalent.

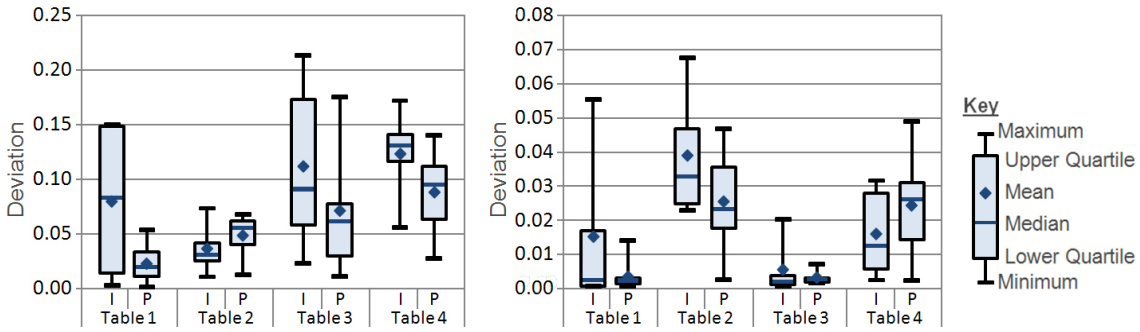


Figure 4. A series of box plots depicting the deviation ratio of participants based on the precision used by the participants

Figure 4 shows the accuracy of the participant's set up. Deviation on the y-axis refers to the difference between the real world value and resulting estimated value divided by the real world value. The left hand graph represents the deviation of calculated distances from real world distance. The right hand graph represents the deviation of the resulting orientation data. The data on each graph is split into groups based on tables then into the subgroups 'I' and 'P'. 'I' stands for imprecise which refers to participants who did not check their inputs. 'P' stands for precise which refers to participants who took their time and were very exact with their inputs. Other behaviour which indicated the precise nature of a participant included crouching to check alignment, moving around each tablet to set the alignment and double checking the alignment before submitting the input. One set of anomalous data was removed from the data used. This set of data was generated from a user who did not understand the set up technique and as a result performed the table set up method incorrectly. The data in Figure 4 shows that the participants who were more precise in their set up achieved a greater accuracy in general. Table location seemingly affected the size of the deviation of distance and table orientation appeared to have an effect on the participants' orientation deviation. The precision of the participants' input does not have as much effect on the deviation of the table orientation as on the deviation of the table distance.

TABLE 2. AVERAGE TIMES (S) TAKEN BY PARTICIPANTS TO SET UP TABLES FOR THE TWO SYSTEMS.

	First Table	Second Table	Third Table	Forth Table	Total Time
Simple System	44.22	27.11	24.78	20.11	128.44
Sophisticated System	28.78	33.00	23.22	18.78	115.00

Table 2 shows the average number of seconds required to set up the tables on both the systems. The table groupings refer to the sequence the tables were set up in and do not refer to specific tables. This is because the order the tables were set up in differed between participants. The total set up time is larger than the sum of the time taken to set up tables due to time taken by the participants to travel between the tables. This discrepancy is larger for the simple system as the participant's also had to spend time identifying which was next in the sequence to set up. For the sophisticated table set up method participants could choose whatever table they wanted to set up next so no time was needed for identifying the next table. As shown in Table 2 a participant's time improves on average for each table set up. This is except between the first two tables using the sophisticated system which could indicate that the positioning of tables could effect the time to set up the tables in the sophisticated method. We can observe from Table 2 that through gaining familiarity with the used method a participant's time improves in general.

The results also showed that the average time taken by the participants to transfer materials between screens was 3.81 seconds on average when using the simple system and 2.72 seconds on average when the sophisticated system was used. This shows that the flick method used by the sophisticated system was on average faster to use than the drag and drop and/or request methods used in the simple system. It is clear that the flick method would be faster than the drag and drop method to use as for only a brief input is required to initiate flick method whereas the drag and drop method needs an input through the whole execution of the method. The retrieval technique which was preferred for transferring materials by exactly half of the participants who used the simple system could be used faster than the flick method. This is because participants could collect materials on one table from all the other tables. However due to the latency of loading information across a network the delay slowed the use of this method down thus resulting in the flick method being the quicker material transfer method on average.

As shown in Figure 5 below participant's ratings of the system elements were all relatively high with little variation between feedback for the two systems. This data seems to support the null-hypothesis that users would rate the sophisticated system lower than the simple system. However all participants who had a chance to use both systems stated that they preferred the sophisticated system. One participant commented that they would prefer a system which used elements from both systems but overall preferred the sophisticated system. Comments from the participants indicated that although elements from the sophisticated system were easier and quicker to use than the alternative methods in the simple system they were rated lower as they took longer to understand which corresponds to the hypothesis. This is backed up by the quantitative data of the timings showing participants' use of the sophisticated methods being much quicker on average than those of the simple system.

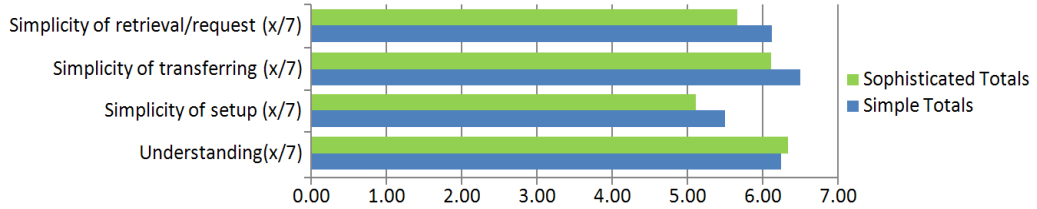


Figure 5. Bar chart of participants' ratings (out of 7) of system elements from the post-session questionnaire

V. EVALUATION

This section contains evaluations on the work carried out during the course of this investigation. The first sub-section focuses on evaluating the solutions selected and the second sub-section assesses the implementation and evaluation of the resulting systems along with the organisation of the investigation.

A. Evaluation of Solutions

The first minimum objective calling for a way of determining the potential configurations of the multi-touch tables was accomplished early on in the investigation. This accomplishment is demonstrated through the simulation tool developed at the start of implementation. This tool allows users to represent any possible configuration of multi-touch tables that is possible in the real world and therefore completes the first minimum objective. The second minimum objective calls for a basic way of setting up tables in response to their layout. This is accomplished by the table set up method implemented in the basic system. The remaining minimum objectives called for a basic method of initiating the transfer of materials from one table to another to be investigated and implemented. The simple system's drag and drop method completed these objectives adequately.

The first intermediate objective called for enhancements made to the method of initiating the transfer of materials from one table to another which utilised realistic physics. The sophisticated system's flick method is the implementation adhering to this objective. Through using the flick motion this utilises the intuitiveness of multi-touch (Kosinen, 2008). As can be seen in Table 2 in the results section this allowed participants to transfer materials quicker than the corresponding simple method. The second intermediate objective stated that a method for manually retrieving documents from other screens should be investigated and implemented. This objective is accomplished by the retrieve method in the simple system and request method in the sophisticated system which both accomplish this objective using similar techniques.

The original investigation specification called for at least one of the advanced objectives to be accomplished, however all three were accomplished to some degree. The first advanced objective called for the investigation and implementation of methods for displaying materials on a vertical display. This is accomplished by the vertical display method implemented in the sophisticated system which can be used for a vertically displayed screen with an edge parallel to the ground.

One of the main focuses of the investigation was the second advanced objective which called for a more intuitive method of setting up tables due to its close relation to the research question. From a close examination of the simple method for table set up it is apparent that the intuitiveness of the system is hampered by the size of the input. With all tables requiring 2 inputs from the user for each set up of transfer areas to all other tables on n tables the size of inputs overall becomes 2n.n-1. This is always the input size and as there is no difference between a best and worst case the input size is of order $\theta(n^2)$ which is a tight bound. However the method used for setting up the tables requires an input size of a much smaller order. As each table will only ever require the 3 inputs from the user the input size per table is fixed and is not dependant on the number of tables. Therefore the overall input size for the set up of tables in the sophisticated system is always $\theta(n)$ which is an improvement over the size of the number of inputs required for setting up the tables on the simple system. Calculations in both systems for processing the inputs are performed in constant time. This implies that the time taken to perform the sophisticated system's trigonometry and transfer area creation calculations is comparable to the time taken for the simple system's transfer area creation calculations. This means despite the sophisticated system's set up method requiring more calculation than the simple system the responsiveness of the system will not appear degraded in the sophisticated system to the user. The constant running time of these calculations also implies that the running time of both systems is entirely dependant on the order size of the input as all other calculations as constant. As shown from the results participants found it more difficult to understand this sophisticated method. However use of the method was quicker and preferred once the participant was used to it. It was noted by participants that the sophisticated made better use of touch technology making it more intuitive. Despite the range of the participants' accuracy of inputs the table set up still returned acceptable location and orientation values in most cases. This indicated that the method can accommodate for user's accuracy to an acceptable extent and is adequately intuitive. The objective calling for an enhanced and more intuitive method of setting up tables can be considered complete by the sophisticated table setup method.

The final advanced objective called for a method of placing constraints on transfers. This objective did not have as much time put into its investigation as other objectives. This is because this objective has little to do with fulfilling the research question as it is actually concerned with the final application of the concepts developed in the investigation in the classroom scenario. However evidence of this objective being adhered to is evident in the implementation of the sophisticated system's material request method. This differs to the simple system's retrieve method as it allows the user on the table which materials are to be taken from to decide whether or not to send the material to the table requesting it or not. This can be seen as a level of constraint over the transfer of materials.

Though not relevant to the objectives networking was necessary for the implementation. The structure and protocol used proved itself sound and only exhibited issues when there were problems with the physical network such as a loss of wireless network signal strength. The only issue which was not satisfactory for the networking was the delay of messages being received. This was likely due to possibly due to the client-server architecture of the network as all messages needed to be routed via the server. This could be improved in future work by

changing to a true peer to peer network and by reducing the sleep time in the network thread update loops.

At the end of the investigation all the given objectives were completed to a more than satisfactory level. The experiments showed the successful implementation of the methods developed and the results demonstrated the design of these methods was more than satisfactory.

B. Evaluation of Implementation

The approach to implementation of this investigation ensured that all objectives calling for an implementation of a particular method and enhanced implementation when called for was accomplished. The use of backups both local and on a remote repository ensured the security of the code during development. I Implementation of the systems was accomplished within the time given for implementation by the investigation's timetable. Overall the organisation of the investigation was virtually issue free due mainly to the use of a well researched plan utilising both hard and soft deadlines. All elements of design and implementation were completed within the predicted time allowing for further improvements to the resulting deliverables. A flexible and in depth design helped identify and correct possible problems early on. This also proved beneficial in having knowledge on what elements of the implementation would be the more prone to issues. This allowed for preparations to be made such as more development or in depth testing for elements of the implementation. As implementation facilitated the accomplishment of the objectives on time and with few issues the approaches taken to implementing the systems can be considered to be successful.

The experiment carried out to evaluate the software developed in this investigation was successful in evaluating both the solutions developed in this investigation and their implementations. The data collected was relevant to the focus of this investigation and was serviceable for evaluating if the investigation's objectives had been met. A large amount of data was collected as part of the experiment of which some was not collated for this report as the data was not wholly relevant to this investigation. This redundant data has been logged for future related studies. Several minor issues apparent in the experiment may have affected participant's confidence in the system. Interference to the wireless network caused communication issues between the interfaces used in some sessions. . Comments from users implied that poor understanding resulted in imprecise inputs and imprecise inputs resulted in an undesirably wide deviation of the calculated values from their corresponding real world value. The explanation of the system use may not have been as clear as needed possibly resulting in a number of results with larger deviation than those that may have been expected with a clearer explanation. The monitoring equipment and software used was largely untested which resulted in a lack of recorded data for some participant sessions meaning timing data for these sessions was not present.

The main limitation of the experiments was the population size which was two small to gain an acceptable confidence level and accuracy for the population of potential users of applications using the methods developed. Another limitation connected to the population of participants was the sampling frame. All the participants were students (of a range of different disciplines) and university computer science research staff which is a threat to external validity. Another major limitation was the use of single touch interfaces where multi-touch would have allowed for a better evaluation of the multi-touch elements of the methods. This is especially true for methods used in the sophisticated system methods which were designed to utilise the possibilities offered by multi-touch. Despite these issues and limitations however enough information was collected to draw conclusions from. The conclusion proved

to be relevant to the scope of this investigation's objectives so the experiment can be considered successful.

VI. Conclusion

The main purpose of this investigation was to the answer the question, "If a multi-touch table has information on its location and orientation in a physical environment relative to other tables, how does this affect user collaboration across all tables?" Through a series of objectives a number of methods were created to answer this question and two systems were implemented to highlight improvements and differences between the methods used. One of these systems used easier to understand methods whereas the other system used more complex methods which made more of the opportunities afforded by multi-touch technology. A series of laboratory experiments then were used to evaluate the software and the concepts behind them to answer the question. In this experiment the system utilising the more sophisticated methods was predicted to be preferred by the participants.

The results from the experiments showed that participants were able to use the methods implemented in both systems with few issues. Comments collected from the participants showed good understanding of the systems' concepts and use. Comments also revealed that the new methods were preferred to traditional methods of transferring materials on computers such as shared network areas used in most operating systems. The methods designed were shown by to the results to have utilised a user's knowledge of the physical environment which answers the research question. Also the results showed that though participants took time to gain familiarity with the concepts involved in the sophisticated system they preferred the use of it after gaining this familiarity. The results showed that the set up of the tables and transferring of materials on average was faster on the sophisticated system. This was what was predicted by the experiment's hypothesis. As the sophisticated is the preferred system of the experiment participant's this indicates that multi-touch improves the intuitiveness of a system. This is because the methods designed for the sophisticated system were designed to make the most of the opportunities afforded by a multi-touch interface.

So with the concepts developed in this investigation proving to have potential, future work involving them either entails finding applications for the concepts or further development of the methods developed. The concepts and methods as they stand at the moment could be implemented into existing or emerging applications. For example the developed methods from this investigation could be could be implemented into the SynergySpace framework meaning any applications developed using SynergySpace could utilise and expand upon these methods. The main issue with the systems was the users' initial understanding of the sophisticated table set up method. Participants in the experiment with greater understanding were in most cases more precise with their input which is more desirable for its use as this returns a more accurate approximation of real world data. As the sophisticated table set up method is the most endearing element of this investigation for future implementation further work on this could entail ways of making the method more suited for easier understanding from its user. This could possibly be achieved through better instructions or even by automating the method through the use of different technologies such as mobile technologies (Kortuem et al., 2005) or visual fiducial recognition such as that used in augmented reality. Also other methods of collaboration using the physical environment other than just sharing materials could be found along with other ways in which multi-touch could be used to improve the methods developed.

It is important to note that while the investigation was started for finding a solution to a specific problem the concepts and methods that were developed for it have many applications

beyond the original scenario. The idea of passing materials between computer interfaces based on spatial information can be applied to almost any computer interface devices and need not be used only in the context of education. The individual methods depicted in this paper could easily be applied for use in many other possible scenarios. Overall this investigation was successful in completing the stated objectives, answering the research question and in adequately proving the concepts and methods developed. Further images and videos relating to this investigation and the use of the resulting systems are available at the author's website (McNaughton, 2009).

REFERENCES

Han, J. Y. (2005). "Low-cost multi-touch sensing through frustrated total internal reflection", in *Proceedings of the 18th Annual ACM Symposium on User interface Software and Technology*, Seattle, October 23 – 26, pp115-118.

Harper R., et al. (2008). *Being Human - Human-Computer Interaction in the Year 2020*. Microsoft Research Ltd.

Karam M., & Sschraefel M. C. (2006). "Investigating user tolerance for errors in vision-enabled gesture-based interactions", in *Proceedings of the Working Conference on Advanced Visual interfaces*, Venezia, May 23 – 26, pp 225-232.

Kortuem, G., Kray, C. & Gellersen, H. (2005). "Sensing and Visualizing Spatial Relations of Mobile Devices", in *Proceedings of the 18th Annual ACM Symposium on User interface Software and Technology*, Seattle, October 23 – 26, pp93-102.

Koskinen, H. M., Laarni J. O., & Honkamaa, P. M. (2008). "Hands-on the process control: users preferences and associations on hand movements", in *CHI '08 Extended Abstracts on Human Factors in Computing Systems,* Florence, April 05 – 10, ACM, pp3063-3068.

Kozma, R. B. (1999). "Students collaborating with computer models and physical experiments", in *Proceedings of the 1999 Conference on Computer Support For Collaborative Learning*, Palo Alto, December 12 - 15, pp39.

Kruger, R., Carpendale, S., Scott, S. D., & Greenberg, S. (2004). "Roles of Orientation in Tabletop Collaboration: Comprehension, Coordination and Communication", *Comput. Supported Coop Work*, **13**(5), pp501-537.

Leat, D. & Higgins S. (2002). "The role of powerful pedagogical strategies in curriculum development", *The Curriculum Journal,* **13**(1), pp71-85.

McNaughton J.A. (2009). *Collaborative Multi-Touch Development*, Available at www.dur.ac.uk/j.a.mcnaughton/local/mmt.htm (Accessed: April 2009).

Nakagawa M., et al. (1999). "A revised human interface and educational applications on ideaboard", in *CHI '99 Extended Abstracts on Human Factors in Computing Systems*, Pittsburgh, May 15 - 20, pp15-16.

Peltonen P., et al. (2008). "It's Mine, Don't Touch!: interactions at a large multi-touch display in a city centre", in *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, Florence, April 05 – 10, ACM, pp1285-1294.

Rogers Y., Hazlewood W., Blevis E. & Lim Y. K. (2004). "Finger Talk: Collaborative Decision - Making Using Talk and Fingertip Interaction around a Tabletop Display", in *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, Vienna, April 24 – 29, ACM, pp1271-1274.

Sharp H., Rogers Y., & Preece J. (2007), *Interaction Design: beyond human-computer interaction*, 2nd edition, John Wiley & Sons Ltd.

Scott, S. D., Grant, K. D., and Mandryk, R. L. (2003). "System guidelines for co-located, collaborative work on a tabletop display", in *Proceedings of the Eighth Conference on European Conference on Computer Supported Cooperative Work*, Helsinki, September 14 – 18, Kluwer Academic Publishers, pp159-178.

Stahl G., Koschmann T. & Suthers D. (2006). "Computer-supported collaborative learning: An historical perspective", in Sawyer R. K. (ed) *Cambridge handbook of the learning* sciences, Cambridge University Press, pp 409-426.

Stewart, J., Bederson, B. B., and Druin, A. (1999). "Single display groupware: a model for co-present collaboration", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: the CHI Is the Limit*, Pittsburgh, May 15 - 20, ACM, pp.286-293.

Tibúrcio, T., & Finch, E.F. (2005). "The impact of an intelligent classroom on pupils' interactive behaviour", *Facilities*, **23**(5), pp262-78.

Multi-Server based Distributed Virtual Environments

Student Name:

Supervisor Nar

Submitted as part of the degree of **Computer Science** to the

Board of Examiners in the Department of Computer Science, Durham University.

Abstract

Context/Background: The idea behind Distributed Virtual Environment (DVE) is to create a virtual environment that avatars controlled by users can walk round in or interact with. But because of the increase of number of avatars, the servers may get overloaded and offer poor response time to users.

Aims: Provide with a suitable partitioning scheme that can transfer the excess workload of one server to others which are not overloaded so that the workload is balanced and all servers can offer reasonable response time to their users.

Method: Servers with lighter workload will have more idle computation resources and bandwidth. Thus they can offer better response time to their users. Hence, we measure CPU use and the amount of data goes through each server to see the workload of the servers. So we can distribute the excess workload of one server to others. And by monitoring the response time to the users, we can compare different schemes.

Results: We applied even, density and adaptive partitioning schemes on the same environment. We found that CPUs use, bandwidths use and the overall response time to users of the environment with adaptive partitioning scheme is better than other two.

Conclusions: The results show that the adaptive partitioning scheme gives the better balance of the workload over the servers and then the whole system can offer better response time to their users.

Keywords – Multi-server based distributed virtual environment, workload balance, adaptive partitioning algorithm.

I. INTRODUCTION

A. Distributed Virtual Environment

Advances in computer graphic, high speed networking technologies and database systems enable computer engineers to create a distributed system which allow users to explore and interact in a three dimensional virtual environment through a local network or the internet (Lui et al., 2002). The virtual environment generally represents the real physical world. It typically composed of high-resolution 3D objects and avatars which represent the users. Users can navigate the avatars in the virtual world. For example, they can walk around and interact with the objects. They can also communicate with other users. Such kind of system is called distributed virtual environment (DVE).

A DVE system helps in many kinds of field such as engineering, medicine, military trainings,

education, and computer/video games. For example, the Olympic Games will be held in London in 2012. Before that, lots of new building for the games or for visitors will be built. Suppose a structural engineer from Durham, a civil engineer from Washington, an artist from Paris, and an investor from London want to have a discussion about the design of the building. With the help of a DVE system, they can enter a 3D virtual environment which represents the real physical London including its buildings and roads at their own homes or offices via the internet. They can walk around in the environment by navigate a 3D avatar, comment on other buildings, show their own ideas by building or removing models and then the new scenes will be visible to others, discuss with others. They can even simulate the real life situations by adjusting some parameters such as the population and weathers.

B. Motivations for Project

Once a user enters the virtual environment, it will keep communicating with the system. Since users may move their positions or interact with the objects and others inside the environment, the system has to updates the statuses of their users. The system will receive and process all updates from users and deliver them to others in the environment. Also, the system needs to handle other works such as synchronization and data security. In practice, however, since the computational resources and bandwidth are limited, when the concurrent number of users in the environment increases largely, the workload of the server will increase significantly (Ng, 2003). The workload will consume the processing power and the bandwidth quickly and thus the performance of the system will be affected significantly. The leads to the poor response time to the users.

The computational resources and bandwidth of a single server is very limited. When the whole DVE system is based on a single server, the power of the server will becomes the bottle neck very soon since the processing power and bandwidth is fixed (Ng, 2004).

Therefore, in general, a large scale DVE system is consists of many servers over a local network or the internet. Each server is responsible for a part of the system. For example, every DVE contains a virtual environment. We can divide the virtual environment into many regions geographically. Then we let each server handle a region. All the users appeared on the region will send their updates or other requests to the server. The server only receives and processes the updates from these users and doesn't need to know any information of the rest of the virtual world or other users. Therefore, in this way, the scale of the DVE system can be much larger than that based on a single server and the system allows many users to explore the environment concurrently.

Multi-based DVE systems are flexible. When the requirements about computer graphic increase or the number of users of the system increases (so that there are more work need to be received by the servers and sent back to the users, or more work need to be processed in the servers) and the current available computational resources and bandwidth are not powerful enough to handle such work in acceptable time, we can re-divide the virtual environment and then let more servers join in. Thus the total power of computational resources and bandwidth of the DVE system increases so that the system can offer good response time to its users.

However, increase the total power of computational resources or bandwidth of the DVE system (for example, increase the number of servers) means increase cost. Moreover, in fact, it is very often that the total computational resources and bandwidth of the DVE system are powerful enough to offer a good response time to users. But the schemes for partitioning virtual environments are not effective enough so that the overall performance is impacted. Partition virtual environment actually means allocate computational resources and bandwidth to handle the requests from a number of

users. A partitioning scheme is not effective enough generally means that it allocates more resources to handle a number of requests from users than those they actually need. And the other way round (it allocates too less resources to handle too many requests). For example, if a DVE system is consists of two servers which are exactly the same and each of both can receive and process 50 requests from users per second. Now there are 100 requests in total from users per second. A scheme that allocates one server to handle 1 request and the other server to handle the rest 99 requests is not effective.

Therefore, how to partition the environment will affect the performance of the DVE system. This paper illustrates an effectively scheme to let the system offer a good response time to the users.

C. Objectives for Project

There are three levels of objectives I can meet.

The minimum objective is to make a simple simulation program in which a DVE system is simulated and some functions such as adjust the number of object/users, adjust the distribution patterns should be available.

The intermediate objective is to implement the adaptive algorithm in the program and also provides with the facilities for measuring the CPU utilization and bandwidth utilization.

The advanced objective is investigate the factors which generate the inter-server communication and then enhance the partitioning method.

In this paper, I've met the all objectives except a part of the third one which is to make a new method. I didn't make a totally new method but modified the original adaptive one to get a better performance.

II. RELATED WORK

A. Communication Architectures of DVE Systems

There are basically two types of communication architectures for DVE systems, peer-to-peer, client-server. Early systems such as NPSNET (Falby et al., 1993), DIVE (Carlsson et al., 1993), and MASSIVE-2 (Greenhalgh et al., 1997) are implemented in peer-to-peer architecture. By applying this approach, all of the users will communicate with each other directly without sending messages to any central server. Thus there is no need of such a central server. Thus the messages among the users can be delivered fast via the network. However, since every user can communicate with all others at the same time, the requirement of the bandwidth is O(N*N). This limits the scale of the DVE system. Although multi-cast communication may be adopted, it is designed to work on LAN only. IP multi-casting can be applied on the Internet but it needs specific multi- cast routers. Moreover, the synchronization problem cannot be maintained in peer-to-peer architecture as well. Therefore, this approach is not suitable to be applied in large scale DVE systems.

To address the scalability issue, systems such as BrickNet (Singh et al., 1995), Community Place (Lea et al., 1997) and MASSIVE-3 (Greenhalgh et al., 2000), are implemented in client-server architecture. With this approach, each user sends message to the server for processing and then the server sends the results to other users or servers. Thus the current state of system can be stored in the system. This helps solving the data consistency problem and recovering the errors of the system. But the main advantage is that the server can filter the irrelevant messages so that the

amount of the messages need to be sent over the network is decreased. However, when the number of users increases largely, the number of messages needs to be handled by the server increases as well. Then the server may get overloaded. Another disadvantage is that the server may become the potential failure point.

As mentioned before, to address this problem, multi-server based DVE system is come out. The virtual environment is partitioned to many regions and we assign each of them to a server. The workload is distributed among the servers and the approach also prevents that a single server becomes the failure point. System such as RING (Funkhouser, 1995), NetEffect (Das et al., 1997) and Citta Tron (Hori et al., 2001) use this approach.

B. Global and Local Load Balancing

Livny and Melman (Livny et al., 1987) have shown that even in a homogeneous distributed system, at least one computer is likely to be idle while others are heavily loaded. Should we make sure that every server always has the same workload, or we only transfer the workload from one server when it is overloaded? This issue involves the loading information that we depends on to do the partition. Currently, there are basically two types of load balancing approaches, global and local load balancing. Global load balancing means that information of workloads of all the servers in the system will be collected to help to partition the environment. In general, in a DVE system, a central server is responsible for colleting the loading information from all servers. Then it determines how to transfer workload among servers and then perform it. As the global load balancing approach considers all the loading information of the entire system, it generally makes more accurate and reasonable decisions. But it has to suffer the delay over the network while collecting the loading information from all servers and then spends computational resources on processing the loading information. It also generate amount of overheads which takes more bandwidth. These advantages affect the performance even more when the scale of the system is large. Local load balancing approach, on the other hand, does the partition only depends on the loading information of the local server. Thus there is no large amount of overheads, no network delay and no need to spend much computational resources on processing the loading information. But it is not as accurate as the global one. For large scale system, in order to maintain good performance for users, the local approach is more suitable than the global one.

C. Area of Interest

Users navigate avatars to explore the environment. However, an avatar doesn't need to know every update of all others. Actually it only needs the updates in the area it can see or hear. Such area is called Area of Interest (AOI) (As figure 1 shown below) (Assiotis et al., 2006).

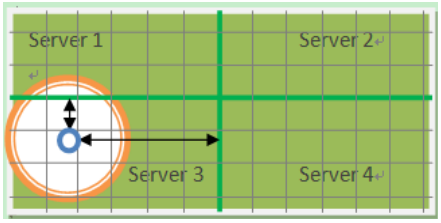


Figure 1. AOI of an avatar. The small blue circle represents an avatar. The Orange circle represent the AOI of it. In this example, part of information of the regions which is hosted by server 1 is conveyed to the avatar.

AOI may base on different relationships between users. For examples, distances between avatars, types of tasks the users are doing or visible areas. Only the information of the objects or users in the AOI of an avatar needs to be sent to that avatar. For example, an avatar moves forward. It sends messages the server hosting it to update its status. The server broadcasts the update to nearby avatars whose AOIs contain it. By using AOI schemes, amount of unnecessary messages to be sent over the network is reduced.

D. Intra-Server Communication and Inter-Server Communication

A user sends requests to the server which is hosting it. The server executes the request and responds to the users. All the messages been sent during the procedure are only between the user and the server. Thus such kind of communication is called intra-server communication for it doesn't involve other servers.

If a server receives a request but it requires information from other servers, the server sends request to the target servers and waits for responses. Then inter-server communications is generated since there are messages been sent among servers. Load migrations belong to inter-communications as well.

With high speed LAN or other networks which have low delays among hosts, the inter-server communications take shorter time than intra-server ones. Hence in this case, we could reduce the intra-server migration and convert them to inter-server. There are some partitioning scheme take this advantages and they will be illustrated soon. However, in general, the status of the network and the quality of the links among hosts cannot be guaranteed to be ideal. If a request requires information from other servers, then the response time to the user becomes longer since more processing are executed. Figure 2 shows the intra/inter communications cause by AOI of users.

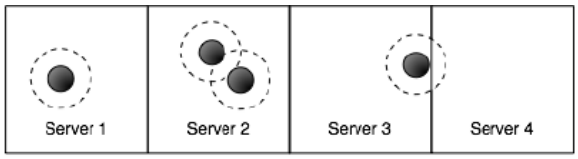


Figure 2. Intra/Inter-server communication caused by AOI. The users in Server 1 and Server 2 only contact with their own hosting server, thus the messages are contributed to intra-server communication. The last user's AOI cross Server 3 and Server 4. Hence inter-sever communication is generated.

E. Existing Partitioning Schemes/Algorithms

In RING, a static partitioning scheme is used. With this approach, the virtual environment is partitioned statically into regions. Each of the regions is assigned to a server. The regions will not be further partitioned during the run time. Hence, when the number of users on a region increases significantly, the server which the region assigned to may be overloaded.

In NetEffect, the DVE is partitioned into zones dynamically based on the density of the users of individual regions. A central server is responsible for checking the density of the clients of zones periodically and organizing the transfer the workload. However, besides the density of the clients, there are other factors such as density of the objects is not considered by this approach.

Since the scheme published in (Reiher et al., 1990), a lot of approaches require access the information of the underlying operating system are introduced. These algorithms basically assess the CPU and get the time that the CPU takes to do the work. The time is treated as the amount of the

work done by the processors (Peschlow et al., 2007). In a cyclic algorithm, a master server checks the loading information periodically. If a server is overloaded, the master server will choose another server which is non-overloaded to make a pair of them. If the difference between the workloads of the two servers is greater than a pre-defined threshold, then the workload migrated between them occurs. The algorithm introduced in (Glazer et al., 1993) is based on the cyclic algorithm. It gathers the information of the CPU allocation and the virtual time advance of processors. These information is used to calculate the amount of work a server should been assigned to. However, this scheme requires homogeneous dedicated systems to balance the load reliably (Jiang et al., 1994).

Greedy assignment algorithm (Ta et al., 2006) shows another way to improve the performance of the DVE system. This algorithm focuses on delay between servers and delay between servers and users. In most of the other algorithms where the client-server architecture is applied, users only communicate with the server which is hosting the user's region. But with Greedy assignment approach, the user chooses the geographical closest server to contact with. This server is called *contact server*. A user only sends requests to its contact server. The contact server may processes the requests locally and responds to the user if the user is inside the region that the server is serving, or it may forward the requests to another server which is hosting the user's region. The delays between the regions and the delays between the users and the servers are investigated. They found that the communication delay between servers is much lower than that between users and servers. By using this approach, the users only contact with the closest server and more messages are conveyed among servers. Hence, overall delay of the system is reduced. However, this approach requires the distributed servers of the DVE system are inter-connected via well-provisioned network links with low network delays and the upper bounds of the delays have been known.

There are several schemes based on the density of the users and virtual environment can be further partitioned dynamically. Quad Tree (Steed et al., 2003) is one of them. With this scheme, the environment is partitioned into regions. The relative density for each region is calculated, and if greater than the threshold the regions is quartered and this step repeats. However, the distributed of the users may not be uniform. Some objects in the virtual environment may be more attractive so that users may congregate in those areas. Simply quartered the region may not give a good partition of the number of users, which means the workload of the new 4 regions may very different. K-d Tree scheme gives a better approach to address this issue. This scheme is similar in nature. When the relative density of a region is greater than the threshold, the region will not be simply quartered but will be dissected by a horizontal or vertical line. The dissection line is chosen so that the two regions have as close to equal total relative density as possible. However, when a region contains a large number of users (which means the workload is heavy as well in general), half of users will be transferred to another server which is large likely to cause the other server get overloaded and this server is too idle. Keep partitioning the region in this way generates high overheads and frequent load migration.

III. SOLUTION

In this section we discuss the design and implementation of the solution. The solution provides with a scheme which solves the load balancing problem and improves the performance of the current DVE systems. First of all, we discuss the architecture and the design of the solution. Then the implementation and other details are presented.

A. Architecture of the Solution

1) Architecture

From the discussion about the architectures in the previous section, we know that with peer-to-peer architecture amount of unnecessary messages are sent over the network so that it requires very wide bandwidth and high-speed links among users. On the other hand, client-server architecture can filter irrelevant messages, which is suitable for large scale DVE systems. Since this paper focuses on large scale systems, we use the client-server architecture in our solution. For the same reason, we give up the single server based system but let the system base on many servers.

When a system is consists of more than one servers, the structure of how the servers combine together affects the way how they work. Share CPU or memory in the system enables any overloaded server accesses the CPU or memory of another server so that more resources will be taken by this server to handle local its own requests. However, this generates more overheads because of the extra inter-server communication. In addition, the data consistency of the shared resources needs to be handled as well. To address this problem, we employ share-nothing architecture (multi-sever support, 10). With share-nothing architecture, one of the servers in the system is often dedicated to be a master server. A master server manages the entire system. It has to know complete data information managed by every other server. Users send requests to the master server. Then the master server forwards it to the server which is managing the data that the requests need. Hence the master server becomes the bottleneck quickly. In order to avoid this problem, we change the role of the master server.

As shown in figure 3, in our solution, we still require a server. However, this server only stores the distribution of the servers over the virtual environment and keeps updating it, so that it knows the hosting server for each region. Moreover, it also knows the basic loading information of each server. We call this server central server.

The central server provides with facilities for many applications of the DVE systems. First of all, it may be used to tell the user which sever he/she should communicate with when the user first enters the DVE system. For example, in lots of online games, users want to appear in the position where he/she left the game last time. With our approach, each user stores the last position before he/she left the game in his/her own host. Next time before the user enters the virtual environment, it asks the central server which server is managing the area containing his/her position. The central server sends back the results to the user, for example, server 3. Then the user starts to contact with server 3 and will not sent any further requests to the central server. Other servers can also ask the central server to identify a region's hosting server's ID. This is only for a server to know which server it should transfer the load to. Hence, the loading of the central server will not be heavy and it doesn't affect the performance of the system. Another weakness with this approach is that the central server may become the failure point of the system. The solution to this problem is to have a backup computer. When the previous server fails the backup computer becomes the central server.

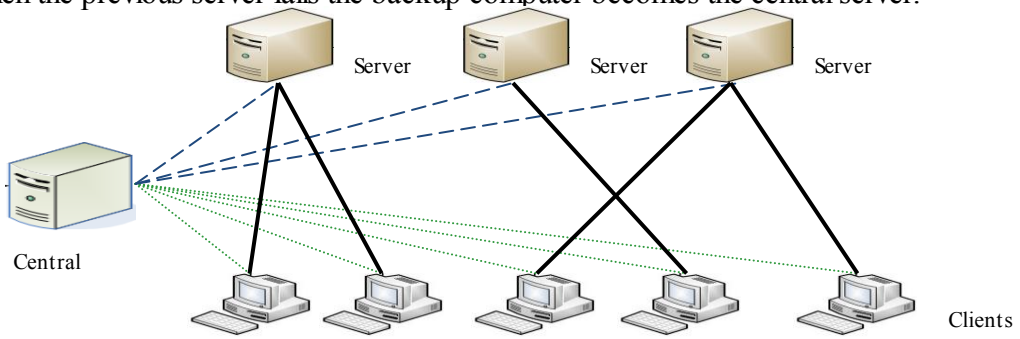


Figure 3. The architecture of the DVE system.

2) Communication

In this paper, to illustrate our partitioning algorithm, the virtual environment is presented by a two dimensional map which is divided into rectangle cells regularly. A set of the cells forms a region. Initially, every region has the same size. Each region is assigned to a server. Then the users can enter in the DVE system. Before they come into the environment, they contact the central server to find out which server is hosting the region they are in. After that, under normal circumstances, the users only contact with it. A server only handles the requests from users who are in the region it is managing.

However, when a user moves to the boundary of a region, the user's AOI may cross the adjacent regions and thus the objects and other users in other regions should be seen by this user. Figure 3below shows different cases of the users near boundaries of regions.

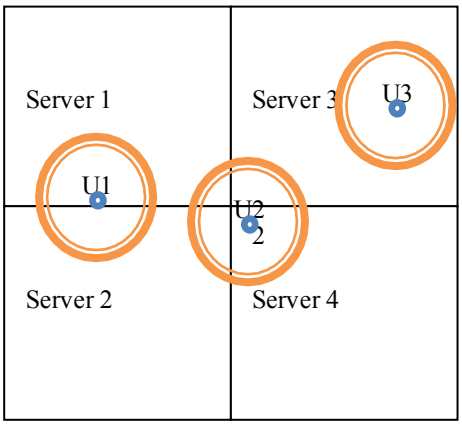


Figure 4. AOI of users near boundary. The small blue circles represent users. The greater orange circles represent AOI of the users.

As shown in the figure 3, the AOI of user 3(U3) is complete in the region managed by Server 3. Hence U3 only contacts with Server 3. We can also see the AOI of user 1 (U1) crosses the two regions managed by Server 1 and Server 2, respectively. We assume U1 was managed by Server 1 only before this and it was moving towards to the region managed by Server 2. When U1's AOI first touches the region managed by Server 2, Server 1 sends a message to the central server to find out which server is hosting the region that U1 just touched. Central server replies Server 1 that the region is hosted by Server2. Then Server 1 sends a message to Server 2 to inform Server 2 that U1's AOI has touched the region. After received this message, Server 2 creates communication with U1 to convey the information of the objects or users in the area that U1 has covered to U1. Now U1 is managed by both of Server 1 and Server 2. When the complete AOI of U1 moves into Server 2 eventually, U1 will stop the communication with Server 1 and only communicates with Server 2. Similar to U1, when U2's AOI crosses three different regions, it communicates with all the three servers.

B. Design of the Solution

1) Load Index

The aim of the project is to let the response time to more users in the system be acceptable. Since

the system is based on many servers, the performance of the system depends on every server. The primary factor which affects the performance of a single server is its load - overloaded servers offer poor response time. Hence we need a scheme to solve this problem.

Assign exactly the same amount of workload to each sever makes every server is always underloaded unless the total workload is beyond the threshold of the entire system. However, this kind of schemes imposes considerable overhead since the any difference among the servers causes the load migration.

Hence, we need a load index to conduct the system when to organize the load migration. This index should not let some server are overloaded but others are idle. At the same time, it also should minimize the overhead imposed during the load migration. In other words, it should be suitable value so that the entire system has a good overall performance. In general, a server offers good response time to its users when it is under its load index. Actually, indentify a suitable load index for a system is the key issue in design of a load-balancing scheme. Currently, there are several load indexes being used such as the length of the CPU queue, the amount of available memory, the rate of system calls, the CPU utilization, and the bandwidth utilization. Kunz (Kunz, 1991) found that the choice of a load index affect the performance of the system significantly. Kunz also shows that combinations of the indexes do not have better performance than that of a single index because of the overhead imposed by using the indexes. Therefore, in our solution, we choose the CPU utilization as our load index in our solution. In order to avoid the bandwidth utilization affect the performance of the server, we monitor the bandwidth utilization as well. However, it is independent of the CPU utilization, which means there is no complex computation of these two values so it doesn't affect the performance. If any of these two value excesses their thresholds respectively, the partitioning algorithm starts. Then we need to monitor the CPU and bandwidth utilization of all servers all the time to determine whether a server is overloaded or not. To make the paper clear, we'll illustrate the algorithm by only using CPU utilization as load index but it is the same algorithm when using bandwidth utilization as load index.

2) Load Monitor

As mentioned before, we employ the local load balancing scheme in our solution. Hence, each server monitors its own loading information. Each server checks its workload periodically and if the workload is beyond the threshold of the server, the region will be partitioned. That is, some part of the region managed by this server will be split out and then be managed by other underloaded servers. In order to know which servers are underloaded, the overloaded server will ask the central server. Then the central server gathers the information of its other servers and then reply. With this approach, two advantages are taken of. First, the central server only asks the load information of other servers when it received the request from the overloaded server. This means the load information is up-to-date. Thus the overloaded server will know the current minimum loaded server it needs. Second, the servers do not have to report their workloads to the central server periodically. Hence the number of messages need to be sent over the network is reduced significantly. In addition, depends on the algorithm we use, the central server may only send the requests to a few servers rather than all others. More details will be illustrated soon.

3) Adaptive Partitioning Scheme

a) Initialization of the DVE System and the Partitioning Algorithm

As introduced in the previous section, there are three types of the partitioning algorithms currently, static, dynamic and adaptive. In our solution, the partitioning algorithm used is adaptive which has more advantages than other two.

By using our adaptive partitioning algorithm, when the DVE system is first started, the 2D map is divided into cells as mentioned. Then the entire map is partitioned into as many regions as the number of servers. Every region has the same size (in terms of cells) and is assigned to a server. Users send requests to the central server to find their own hosting servers and then start to communicate with it.

Each server only knows the cells it is managing but not others in the virtual environment (VE). In our solution, each server keeps an array of arrays which represents the 2D map of the VE. If a cell is managed by the server, its value is 1. Otherwise is 0. It only handles the requests from the users on its region. Every server keeps 4 boundary points of the region it is serving. The 4 boundary points are actually the 4 corner points of the smallest rectangle which can contain/cover the region that the server is managing. For example, figure 5 shows the 4 boundary points of a region. In figure 5, all the blue cells are managed by a server. The 4 grey cells are its boundary points. If we draw lines among the points as we did in the figure, we will get a rectangle whose corners are the 4 grey cells. This rectangle is the smallest rectangle which can cover all the blue cells although some of the cells in the rectangle are not managed by this server. Note that the cells except the blue ones are not managed by this server. Also note that the 4 grey cells are not hosted by this server as well in this case.

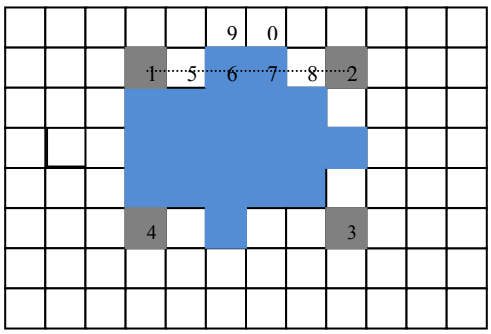


Figure 5. The boundary points of a region.

Each server waits for the requests from its users. It may process the requests, filter the irrelevant messages and then broadcast the results to other users or may send requests to the central server, which depends on the situations as mentioned before. Each server checks its own CPU utilization periodically. If the CPU utilization is beyond its threshold, then this server starts to partition its region.

b) Selection of Target Servers

In many distributed systems other than DVE systems, an overloaded server may transfer its load to any other servers obviously including the lightest loaded server in the entire system. However, in DVE systems, because users will receive updates from the object or other users in their AOI and we aim to reduce the inter-server communication as much as possible, it is reasonable to let the areas that a server is managing as concentrated as possible. Hence, in our algorithm, an overload

server offloads some of its load to its neighbor servers. To finish this task, first of all, the overloaded server needs to know which server it should offload its load to. We call that server the target server. In our solution the target server should be the lightest loaded server of all its neighbor servers. Now we need a scheme to select it.

First, the overloaded server finds whether the cells on the 4 boundaries of the rectangle (which is formed by the 4 boundary points as mentioned above) are managed by it or not. If a cell on the 4 boundaries is not managed by it, then it ignores the cell and continues for next one. If the cell is in its region, then it will send a message to the central server to ask which server is managing the cell's neighbor cell which is not in its region. The central server has the knowledge because it maintains the latest distribution of the servers on the map as discussed above. These steps repeat until all of the cells on the 4 boundaries have been checked.

For example, if the server which is hosting the blue cells in figure 5 is overloaded, first, it checks whether the cell 1 is managed by it or not by taking advantage of the array of arrays as mention above. It will find the cell 1 is not under it control and then it checks next cell which is cell 5. Cell 5 is still out of its region then it checks cell 6. After it found cell 6 is managed by it, it sends a request to the central sever to ask which server is hosting the cell 9 as well as the loading information of that server. The central server sends the result back. If the load of this server is under it own threshold, the overloaded server will simply store the result since this is the first neighbor server it found.

This repeats for cell 7, thus it ask will about the server information of cell 0. But this time when it receives the server ID which is hosting cell 0 as well as its loading information from the central server, it will compare the new result with the old one which is the sever information of cell 9. It may take 2 steps to compare them. First, it compares the IDs of the two servers which are hosting cell 9 and cell 0, respectively. If the IDs are the same then it ignores the new one. If the IDs are different, the loads of the two servers are compared and only the ID and the load of the server with less load is maintained. For example, if cell 9 is managed by Server 1 and cell 0 is managed by Server 2, and Server 1 has heavier workload. Then the information about Server 1 will be removed. The Server 2 and its load will be kept. Hence, eventually it will know the lightest loaded server.

c) Target Load

After indentified the target server, the overloaded server needs to know how much workload it need to offload in total as well as how much for a target server. We call the amount of the workload target load. In order to maintain the performance of the system, the target load should satisfy two conditions. First, the overloaded server should be underloaded after offloaded such amount of load. Second, the overloaded server should not be overloaded again in a short time under normal circumstances, which depends on the applications. In our solution, an overloaded server keeps offloading its load until its load is no more than the 85% of its threshold of the load index.

d) Partitioning Algorithm

This subsection describes how the target load of an overloaded server will be transferred to the target server. We know each server manages some region(s) and it handles all the requests from the users on the region(s). By removing a part from its original region will reduce the area that the server is managing. Hence, the requests from the users will be reduced and so is the load. In our solution, basically, we offload the load by transferring some part of the geographical region which is corresponding to the cells on the boundary of the region from this server to another. Details are

illustrated by using the example in figure 6.

In figure 6, we assume that the region which is consists of blue cells is managed by Server 1, the region which are the green cells is managed by Server 2, and the red cells are managed by Server 3. White cells are not managed by any of these three servers and we don't consider their servers for now. The horizontal axis is x-axis and vertical one is y-axis.

We assume Server 1 noticed that it is overload. Both of Server 2 and Server 3 are underloaded and Server 3 has lighter load. We also assume Server 1 has found the target server which is Server 3 and it stores the Cell F through the method we discussed in the Target Serer Selection section. Then it starts to transfer some of its cells to the target server.

First of all, Server 1 sends a message to the Server 3 to let it know Cell F now is managed by Server 3. Then Server 3 adds Cell F into its own region. After that, Server 1 sends a message to the central server to update the distribution of the servers in the central server, which means Server 1 let the central server know that the Cell F now is managed by Server 3. Finally, Server 1 removes Cell F from its own region so it will not handle any requests from the users on the Cell F.

Then the Server 1 checks its load. If the load is more than 85% of its threshold, then the Server 1 checks two cells, fist, whether the next cell - Cell G is in its region or not. Second, whether the adjacent region of the Cell G is managed by Server 3 or not. In Figure 6, both answers are yes. Thus the transferring steps above repeat for Cell G. Server 1 will continue transferring the cell on the boundary until any of the following third cases happened.

Case 1, after transferred a cell to Server 3, Server 1's load is no more than 85% of its threshold.

Case 2, the next cell is not managed by Server 1 which means either the next cell is out of the boundary or it managed by another server.

Case 3, the adjacent region of the next cell is not managed by the target server which is Server 3.

For case 1, the algorithm simply terminates. For the last two cases, after the algorithm terminated, the boundary points may be modified by Serve 1 and Server 3. Server 1 checks whether there is any cell on this boundary. If there is, then Server 1 does nothing. But if there is no cell, Server 1 will replace Cell B and Cell J by Cell L and Cell M, respectively. Server 3 will check whether its two top boundary points has the same y-value of Cell B or J. If they are, Server 3 does nothing. If they are not, the y-value of the two points will be decreased by 1. After these, Server 1 will go back to the fist step to find another target server and the steps repeat until case 1 happens and then the algorithm terminates.

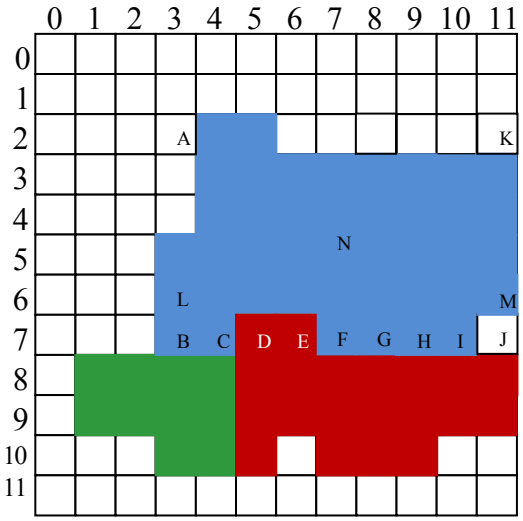


Figure 6. A snapshot of a DVE system.

However, data consistency errors maybe happened because of the latency of communication. For example, if Server 3 is overloaded and it will transfer Cell D to Server 1. But just before this, Server 1 is found to be overloaded and it transferred Cell C to Server 2. Then actually Cell D will be split to Server 2 rather than Server 1. The Server 3 doesn't know this, which will generate data consistency error. In order to void this in our solution, the overloaded server can split cells to only one server at a time. The target server can only receive cells from the overloaded server and cannot split any cells out.

Basically, the main idea behind the algorithm is that the overloaded server identifies the lightest loaded server of all its neighbor servers as the target server. It then split its cells on the boundary to the target server so that the load will be decreased. The reason why an overloaded server transfers the cells on the boundaries only to the target server is because the AOI of users may cross different regions which are hosted by different servers. For example, when Server 1 is overloaded and it transfer Cell N to Server 2. Then the possibility that users in or near to Cell N can see objects or other users in different regions is increased. That is, probably more users will be served by both of Serve 1 and Server 2.

C. Implementation

a) The DVE system

We developed a simulation program and run the experiments on it too see the performance of the algorithm. Since the simulation program is not a real software application, some factors such as the quality of the graphics are not concerned. However, we modify some other factors to make it exhibit the performance of our algorithm in order to compare with other algorithms.

We use a 2D map to represent the DVE system. The map is divided into 30 * 30 grids. There are nine servers in total and each of them has a 10 * 10 region initially. Figure 6 shows this below. Regions with different colours are managed by different servers. The white dots are objects. The bigger white disks are users.

The load index in the program is CPU utilization and the threshold for each server is 25% of the total CPU time. However, sometimes the CPU time for a server is beyond the threshold for a short time but it does not worth to partition the region since it generate more overhead. To avoid this problem, the server checks two parameters. The server first checks the CPU time, if the time is over the threshold then the server checks whether the number of users is more than the threshold of users which 100 in our simulation program. If the number of users doesn't reach the threshold, then server do nothing. If both are over the thresholds, then the partitioning algorithm starts.

As mentioned in section B. (2), the same partitioning scheme starts when the bandwidth utilization excesses its threshold as well. Each server stores two counters, one for intra-server messages and the other for inter-server messages. When a message is received or sent, the server determines which type the message is and then increases the corresponding counter.

b) Users

A user navigates an avatar to explore the VE. In the simulation program, we use a disk to represent a user. A disk move randomly and updates its position every millisecond. An avatar's AOI is a 3*3 grids and it is in the central cell.

c) Experiment

Since the patterns of users' distribution may be either uniform or skew, we need to study both of the patterns. The simulation program provides with two functions. One is let the avatars distributed uniformly. For the other one, we need to point the central cell and the side of the area, and then the avatars will be concentrated in a given area. Figure 6 shows the patterns. The white dots are objects. The bigger white disks are users.

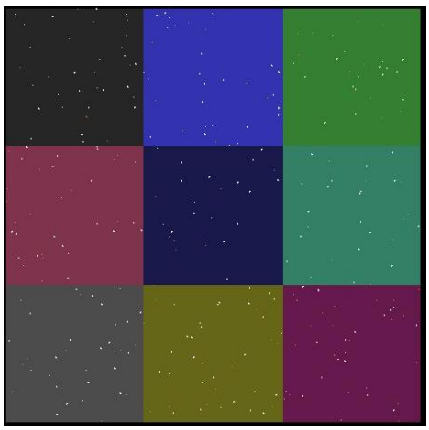


Figure 6(a). Uniform pattern.

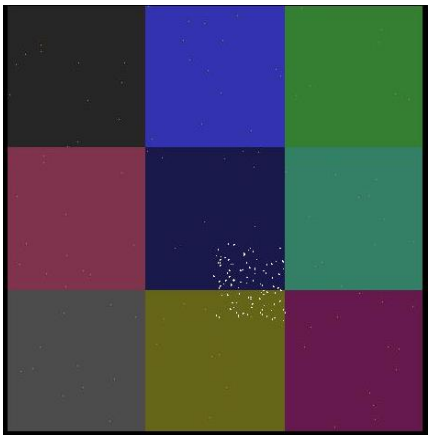


Figure 6(b). Skew pattern.

In the experiments, we implement another two algorithms which are static even partitioning algorithm, Quad Tree algorithm and then compare their performances with our algorithm. By the performance we mean the response time to the users. The implementations of the algorithms in our experiments are basically follows their descriptions in this paper earlier, respectively.

IV. RESULTS

We devise three experiments in total to see the performance of our scheme and investigate the factors which can affect the performance. Each of them is designed to see some particular aspects of the performance.

In order to make the result as convincible as possible, we run some number between 30 and 50 of times to get each sub-result. And in total we run over 1500 times for all experiments.

A. Experiment 1

In this experiment, we investigate the affection of the number of servers on the performance of a DVE system. Basically, we change the number of servers for each test but keep all other factors fixed (the "test" in this section means a component of the experiment. An experiment includes several "tests"). This experiment is consists of 4 tests and we run each of the tests for 30 times. Then we remove the highest and lowest values and get the average one.

The numbers of servers are 1, 3, 9 and 36 for each test, respectively. There are 500 objects and 200 users in the system. The distribution pattern for the objects and users is uniform. The partitioning algorithm is the static even one as we mentioned in the second section. Figure 7 shows the result.

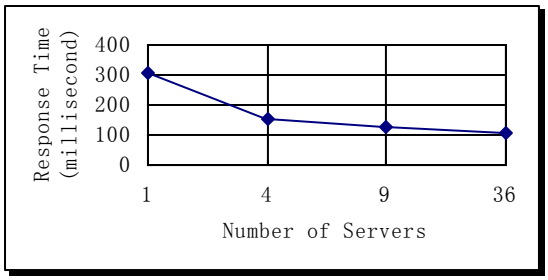


Figure 7. Effect of number of servers on the performance.

We observe that the performance improved dramatically along with the decrease of the number of the servers. We assume 200 milliseconds is acceptable for users. When there are 200 users, a single server based system will be overloaded.

B. Experiment 2

In this experiment, we study the effect of number of users and number objects on the system. We fix the number of servers at 9 in the experiment. The number of objects varies between 500 and 4000. The number of user is between 100 and 800. Every test has a different number of users or different number of objects. The distribution pattern is uniform. The static even algorithm is used.

For each test we collect the average response time and the number of users distributed on different range of response time. Figure 8a and 8b below show the results of average response time. Figure 10 shows the result of users' distribution.

From the figure 8a and 8b we observe that when the number of objects is fixed, different numbers of users give very shows performance. However, when the number of users is fixed, different numbers of objects do not affect the performance significantly. Hence, we see the number of users have much more impact than the number of objects on the performance. It is also clear that the response time is longer than the acceptable time if there are more than 800 users.

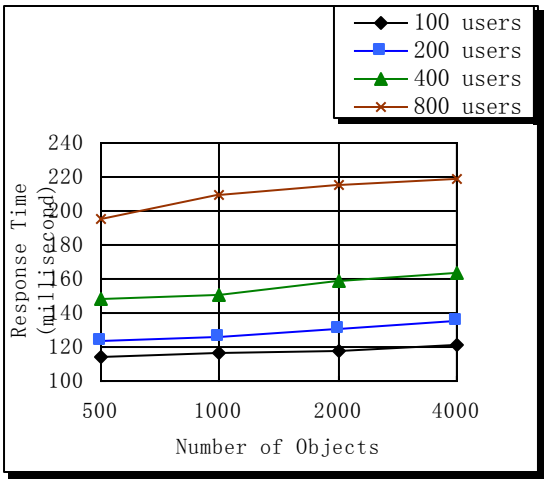


Figure 8(a). Effect of number of object.

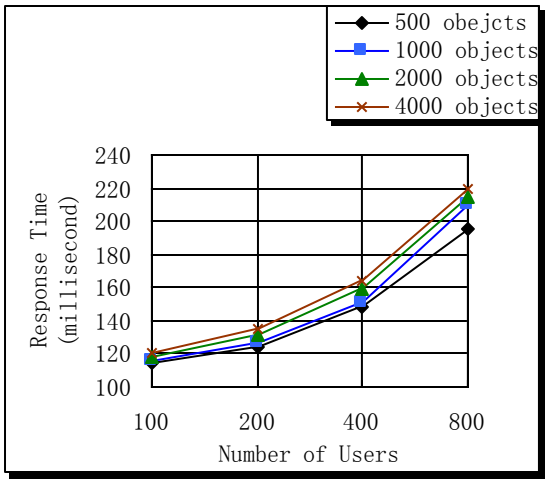


Figure 8(b). Effect of number of users.

From the figure 9 we can see that by using uniform pattern and even scheme, when the number of users is lower than 400, the system have a good performance. But when the number is over 800, the performance immediately becomes very bad. Again, we can also observer that the objects doesn't affect the performance a lot.

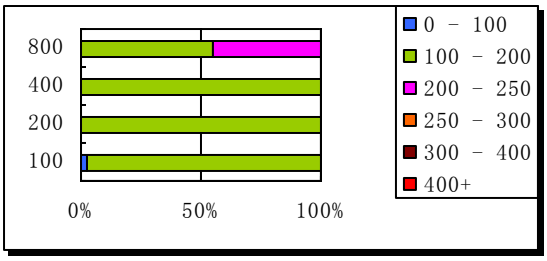


Figure 9(a). Users' distribution with 500 objects.

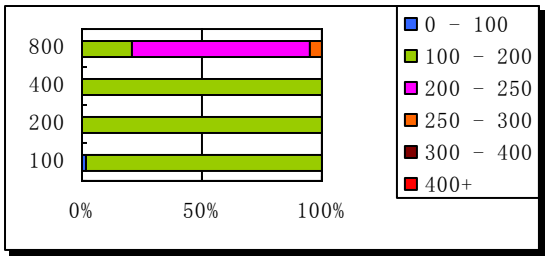


Figure 9 (a). Users' distribution with 1000 objects.

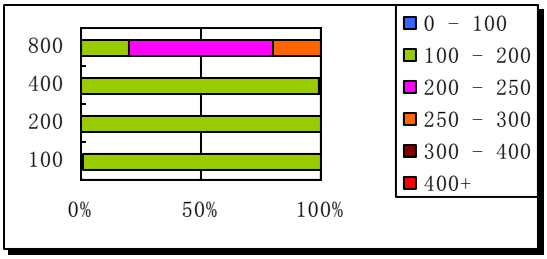


Figure 9(c). Users' distribution with 2000 objects.

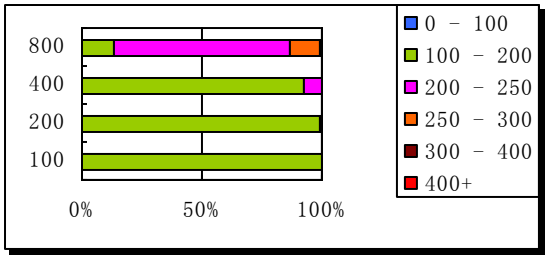


Figure 9(d). Users' distribution with 4000 objects.

C. Experiment 3

In this experiment, we study the effect of different algorithms on the performance of the system. The number of servers is fixed at 9. Since we have seen that the number of objects doesn't impact the performance significantly, we fix the number of objects at 500. We compare three partitioning algorithms, static Even algorithm, Quad Tree algorithm and our Adaptive algorithm. For each test, 10% of total users' distribution pattern is uniform. 40% of the rest, which is 36% of total users are concentrated on only one server. The rest, which is 54% of total are in another single server. Again, we collect the average response time and users' distribution corresponding to response time. Figure 10 and figure 11 show the results.

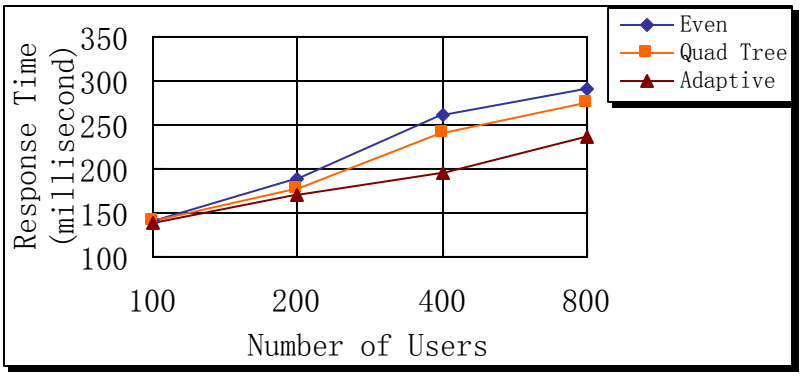


Figure 10. Average response time.

From figure 10 we observe that the adaptive algorithm improve the average response time to the users. We also observer that along with the increase of the users, the differences become more obvious.

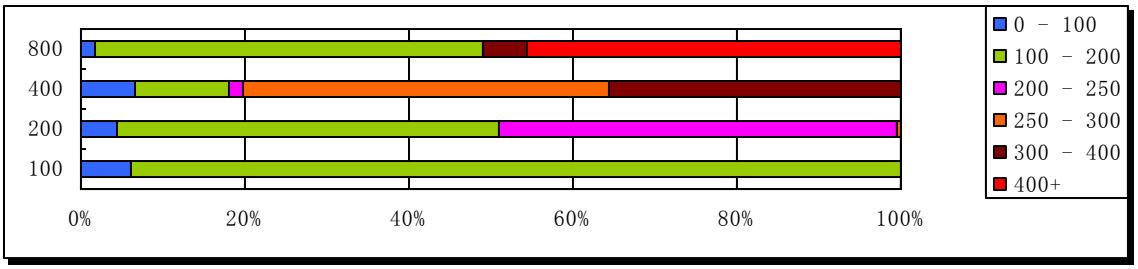


Figure 11(a). Users' distribution using Even algorithm.

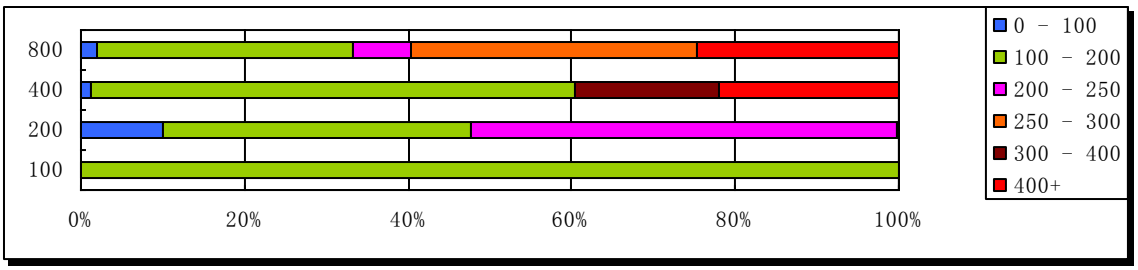


Figure 11(b). Users' distribution using Quad Tree algorithm.

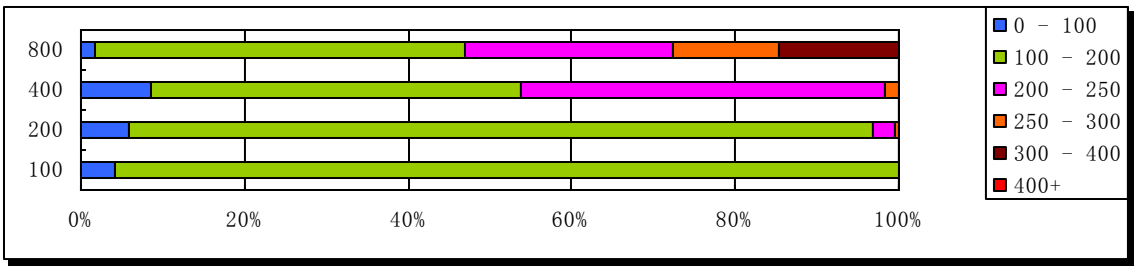


Figure 11(c). Users' distribution using Adaptive algorithm.

From figure 11, we observe that, compare with Even and Quad Tree algorithm, the adaptive algorithm not only reduce the average response time but also let the system offer a much better response time to amount of users. For example, in figure 11(a) although about half of users' response time is less than 200 milliseconds but there are almost the same number of users' response time is longer than 400 milliseconds. However, in 11(c), there is no user having more than 400 millisecond response time although not many users have very short response time.

Figure 12(a) shows the situation while the partitioning is doing. Cells are split from the servers and thus the regions are changed, not square any more compare with figure 12(b).

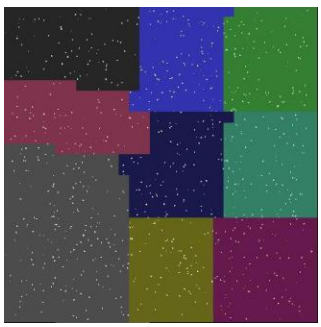


Figure 12(a). Partitioning algorithm is running.

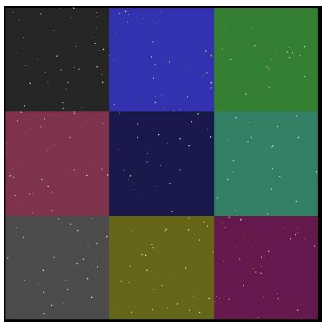


Figure 12(b). No server is overloaded.

V. EVALUATION

From the results in last section, it is easy to observe that our adaptive algorithm can improve the overall performance of a DVE system. Actually, many papers about DVE systems claim that they

have different algorithms which can improve the performance. However, by performance they mean the average response time to the users usually and few of them consider other measurements. Hence, some algorithms can decrease the average response time but does not have a good effect in practice. For example, for 400 users, the Quad Tree algorithm above had many users with more than 400 milliseconds response time, which Even algorithm did not have at all. This is because the Quad Tree had to partition the regions frequently so that it had high overhead. The users in such areas would been affected. However, with our adaptive algorithm, we found that the improvement of the performance was reflected not only in average response time but also in the percentages of users distributed over some response time range. And this can be further taken account in other partitioning algorithms developments. The performance was due to we divided the map into cells which was more accurate while partitioning and we let as few server overloaded as possible. We didn't have many users with very long response time, which means users' response time in our algorithm was closer to the acceptable one. In practice, the shorter the time differences are, the harder the users can realise. Hence, with our algorithm the system can actually show a better performance.

We used many methods to improve the maintainability of DVE system, which actually could also be applied in the future while developing DVE systems. For example, besides the scalable architecture, we also investigated the inter-server and intra-server communication and the factors which can affect them. So in other algorithms, people can adjust the proportion each communication taken by modifying these factors' values. For example, the greedy algorithm mentioned in the second section has shown to be effective but it requires high speed links among servers. By knowing and modifying the factors, its performance probably could be further enhanced. For each single server, we reduce many non-necessary values it has to store and update. For example, we use the boundary points to monitor the regions and also to do the partitioning. Compare with store all edges or all points, it has 4 advantages. First, it reduces the parameters need to be stored for each server. Second, it also means the time is saved while looking for a value, especially when the checking is very often. Third, it also reduces the overhead for updating them. Fourth, they are easy to maintain in the future.

The central server is helpful and not costly. As discussed already, other servers can ask the servers' distribution over the environment and the also the loading information of the servers. The first time a user enters the system, the central server will forward it another server. For the first function, the central server only needs to store array of arrays to match the cells to integers. For the second, it only update loading information when it receives requests which is not many again. For the third, it's only receives the requests from users for the first time so it's not many again. Moreover, each of the three functions can actually be run as an independent process on different servers which are not dedicated to be a central server. Then other server may ask them to get the results. This reduces the cost of the system and also prevent from the failure point, and at the same time it provides useful functions.

However, there are also some drawbacks and limitation. First of all, we use java to create the simulation program. It is not only slower than C/C++ but also doesn't have any facilities to get the information of the underlying system directly. Hence we have to get the CPU run time for each thread first by using some existing API class and then further calculate to get some other values. Although we run the tests for many times, this problem still cannot be avoided completely. Hence the values we achieved may be not very accurate. I didn't use C/C++ simply because I am familiar with them very much. But I think C/C++ is more suitable for this project. Second, I use JME libraries to support the scenes to display on screen. JME is much faster than Java 3D especially when there are

many objects need to be run at the same time. However, it is not as convenient as OpenGL. I got in trouble when I was trying to export the simulation program because the jar file cannot run without JME libraries and most the underlying systems do not have JME built in. If we simply export the jar file from the Eclipse which is my development tool, the jar file even cannot run on my computer. We have to set the libraries. If I could re-do this project, I'll choose OpenGL rather than JME. Another limitation is that the simulation program is not a real DVE system. That means, the data we've achieved from the experiments are not the same as a real system. Factors such as the quality of the links and distances between users and servers, the stabilities of the servers, and quantity of the real messages sent by users will be very different and more complicated than those in our experiments. This problem can be avoided only we have a real DVE application and many real users.

In order to meet the objectives of this paper, we developed an algorithm in this paper. Before that, we first analyzed the DVE systems to find which aspects can be improved significantly and practically. We found there are many factor may affect the performance such as the communication architecture, the balancing problem and the partitioning scheme. It's quite obvious the partitioning algorithm is the part we need to concentrate on. Then we create developed the algorithm and take experiments on it.

VI. CONCLUSION

In this paper we can found that the algorithm can affect the performance of a DVE system directly. In order to show a good performance, it is important that we don't let any server overloaded except it's impossible to be avoided. Hence no matter what the algorithms are, they always need to transfer some of workload from one server to another. In other words, balance the workload among different server will always be the key issue to improve the performance.

The ideas in our paper such as we divide the map into cells, using of points and measure the CPU and bandwidth utilization can be further used in other development of the DVE system.

Almost all the current existing partitioning algorithms only transfer the workload after a server was overloaded. Actually, for the further work, we can see the load balancing from another perspective. We can predict where the workload is going to. In other words, we can predict which server will be overloaded soon and we transfer either some of its original workload to other servers or we can also transfer some of the coming workload to others. In either way, we will prevent the server from overloaded. The advantage of this technique is, the users in the system will not feel the response time becomes long simply because there is always no server is overloaded.

Finally, it is also useful for test the algorithms by using a real DVE system because some unexpected factors may be affect the algorithms a lot. It is also important to use the statistical methods to collect and analyze the results. For example, the percentage of the users who can receive the response time rather than a simple average response time.

REFERENCES

[1] John C.S. Lui & M.F.Chan (2002). An efficient partitioning Algorithm for Distributed Virtual Environment Systems, IEEE Transactions on parallel and dsitributed systems, VOL.13, No.3, 2002, IEEE.

[2] Beatrice Ng, Frederick W.B.Li, Rynson W.H.Lau, Antonio Sin, Angus Siu (2003). A performance study on multi-server DVE systems, 2003, Elsevier Science Inc.

[3] Beatrice Ng, Rynson W.H.Lau, Antonio Si, and Frederick W.B.Li (2004). Multi-Server Support or Large Scale Distributed Virtual Environments, 2004, IEEE.

[4] J. Falby et al.(1993). NPSNET: hierarchical data structures for real-time three-dimensional visual simulation, Comp.Graph.17(1)(1993)65-69.
[5] C. Carlsson, O. Hagsand (1993). DIVE-a multi-user virtual reality system, in Proceedings of IEEE VRAIS, 1993, pp. 394-400.
[6] C.Greenhalgh, S.Benford(1997). A multicast network architecture for large scale collaborative virtual environments, Proceedings of ECMAST' 97, 1997, pp. 113-128.
[7] G. Singh, L.Serra, W.Png, A.Wong, H. Ng(1995). BrickNet: sharing object behaviors on the net, Proceedings of IEEE VRAIS, 1995, 99. 19-27.
[8] R. Lea, Y.Honda, K. Matsuda, S.Matsuda(1997). Community place: architecture and performacen, Proceedins of the VRML'97 Symposium, pp. 41-50.
[9] C. Greenhalgh, J. Purbrick, D. Snowdon, (2000). Inside: MASSIVE-3: flexible support for data consistency and worldstructuring, Proceedings of CVE'00 2000, pp. 119-127.
[10] T. Funkhouser (1995). RING: A client-server system for multi-user virtual envrionments, Proceeedings fo Symposium on Interactive 3D Graphics, , pp. 85-92.
[11] T. Dast et al(1997). NetEffect: a network architecture for large-scale multi-user virtual world, Proceedings of ACM RST, 1997,pp.157-163.
[12] M. Hori etal (2001). Scalability issues of dynamica space management for multiple-server networked virtual envrionments, Proceedings of IEEE Pacific Rim Conference oon Communications, 2001.
[13] M.Livny, M. Melman (1987). Load Blaancing in Homogeneous Broadcast Distributed System, ACM Computer Network Performance Symp. 1987, pp. 47-55.
[14] Marios Assiotis, Velin Tzanov (2006). A Distributed Architecture for MMORPG, Netgames'06, October 30–31, 2006.
[15] P. L. Reiher and D. Jefferson(1990). Virtual time based dynamic load management in the time warp operating system. In Proceedings of the 1990 SCS Multiconference on Distributed Simulation, pages 103–111. SCS.
[16] Patrick Peschlow, Tobias Honecker, Peter Martini(2007).A Flexible Dynamic Partitioning Algorithm for Optimistic Distributed Simulation, Principles of Advanced and Distributed Simulation (PADS'07), IEEE.
[17] D.W. Glazer and C. Tropper(1993). On process migration and load balancing in time warp. IEEE Transactions on Parallel and Distributed Systems, 4(3):318–327.
[18] M.-R. Jiang, S.-P. Shieh, and C.-L. Liu (1994). Dynamic load balancing in parallel simulation using time warp mechanism. In Proceedings of the 1994 International Conference on Parallel and Distributed Systems, pages 222–229. IEEE Computer Society.
[19] Duong Nguyen Binh Ta, Suiping Zhou and Haifeng Shen(2006). Greedy Algorithms for Client Assignment in Large-Scale Distributed Virtual Environments, Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation (PADS'06), IEEE.
[20] Anthony Steed, Roula Abou-Haidar (2003). Partitioning Crowded Virtual Environments, ACM, VRST'03, October 1-3, 2003, Osaka JAPAN.
[21] T. Kunz (1991). The Influence of Different Workload Descriptions on a Heuristic Load Balancing Schemem, July 1991, IEEE.

Point Cloud Data Geometry Processing

Student Name:
Supervisor N:

Submitted as part of the degree of **Computer Science** to the

Board of Examiners in the Department of Computer Science, Durham University.

Abstract –

Context: In recent years, the rapid development of desktop computers with dedicated graphics hardware means the standard desktop is affordable with high performance. Point clouds are being more commonly used in 3D graphics and visualisation applications. Terrestrial and airborne laser scanners used by geologists capture large point clouds that then need to be processed and analysed, often manually.

Aims: To develop and study the efficiency and scalability of an algorithmic tool for the highlight and detection of features in reconstructed and triangulated point cloud data surfaces, with a focus on surface variation.

Method: In order to estimate the surface variation, vertex normals are estimated and compared to each other to obtain an approximate gradient variation for that surface. Depending on how much the gradient changes, we can detect which areas are most likely to be interesting and therefore guide the identification of meaningful features. Performance tests are then run in order to optimise the code and algorithms. OpenGL is used to display the mesh structure.

Results: We found that our algorithms run in linear time even on very large meshes, but that the memory usage was quite significant. Displaying those large meshes graphically was much slower than expected.

Conclusions: The results show the limitations certain components of a programme can force onto itself. The algorithms are relatively efficient, but applying the results graphically proved quite difficult due to the data structure which seemed to act as a bottleneck.

Keywords – Point Cloud, Mesh, Normal, Gradient, OpenGL.

I. Introduction

A. Point Cloud Data

When looking at 3D graphics and visualisation, point clouds are widely used. Developments in laser range scanning technologies such as LIDAR scanning (**Li**ght **D**etection **A**nd **R**anging) or TLS (**T**errestrial **L**aser **S**canning) have seen a rise in the use of virtual rock data. Geologists and seismologists can now make precise airborne or terrestrial scans of terrain. Because laser scanning provides data with high geometrical accuracy and spatial resolution, it can be used to study the outcrops and extract meaningful information from them. The output of the scan is a point cloud consisting of thousands to millions of laser distance measurements that represent the scanned surface. Used in combination with a high-resolution

digital camera, the points can be coloured by applying texture-mapping techniques which makes the dataset much more readable and interpretable. To date, this interpretation is mostly done manually and it still remains a challenge to extract features from those point clouds (Kurz, et al., 2008).

Indeed, these datasets do not provide any information about the relation between one specific point and any of the other points. This obviously keeps the data structure simple, therefore requiring less storage space since including any other single piece of information for each point instantly increases the memory demand by the total number of points in that particular scan or set. As a result, reconstruction and estimation of the relationship between points is necessary. A very common method of recreating a surface is by segmentation and triangulation, such as Delauney triangulation, of the set of points.

With the increase in power of consumer computers and dedicated graphics hardware and software, 3D visualisation of geological data, which was commonplace in petroleum companies, is now affordable for field geologists who still greatly rely on standard methodologies for recording and sharing observations that are mostly 1D or 2D (Jones, et al., 2007; White & Jones, 2008).

B. Surface Gradient

There are many definitions of a gradient. They range from it being the rate of inclination of a slope (its steepness) to much more general definitions such as the amount of change with regards to distance of a quantity pointing in the direction of maximum rate of change. This project does not really attempt to calculate the gradient precisely, but focuses more on measuring the magnitude of change in a surface efficiently and tries to show that rate of change visually.

For our project we will define the gradient change as follows. In a mesh structure, if all neighbour vertices of a target vertex have a normal vector that is parallel to that target's normal vector, the gradient change will be zero, which means that the surface around the target vertex is *flat*. If the surface is uneven, the neighbouring vertices will have normal vectors pointing in different directions. The gradient change will therefore have a value that will be small if the surrounding surface does not vary much, and will be great the more it moves away from being flat. It has similarities to curvature which can be defined as the amount by which a line of geometric object diverges from being flat or straight.

C. Project aims

The aim of this project was to design and implement an algorithmic tool for the estimation of surface variations of point cloud data collected from laser range scans, followed by the evaluation of the efficiency of the constructed application. The project made an attempt to study the possibility of designing a tool that could help detection of features in geological rock scans through gradient calculation/estimation within point cloud data.

To achieve this, the project was split into three main objectives: minimum, intermediate, and advanced objectives, which we outline below.

Minimum Objective – The minimum objective of this project was to design and build a programme that would display a polygonal mesh of triangles, with help of a mesh data structure. The basic mesh data structure would have to enable future features that the next objectives would require.

Intermediate Objective – The intermediate objective of this project was to create and incorporate the algorithms for the calculation of triangle surface normals, vertex normals and gradient (following our definition) in the initial programme, and enable visualisation of the "interesting" gradient changes and vertex normals on the mesh displayed by the programme.

Advanced Objective – The advanced objective of this project was to evaluate the speed of the algorithms and compare it to the time complexity that we obtained from theoretical analysis of the algorithms. The final sections of this paper will show the results acquired and the analysis and evaluation of the work we carried out.

The remainder of this report will document the related work that was examined during the project, the proposed solution, the results collected, an evaluation of the results and the project, and a conclusion discussing the importance of this project and possible future work.

II. Related Work

A. Vertex Normal Computation

The method we chose for gradient estimation (following our definition) is not possible without vertex normals. Max Wagner's paper (Wagner, 2004) covers this issue with the aim to create a smoother appearance when lighting and visualising a mesh. He explains that strictly speaking, a vertex cannot have a normal on its own. Flat surfaces have normals because they are the vectors that are perpendicular to that surface. The author looks into different methods of calculating the vertex normals in order to "enhance" the smooth appearance of a lit mesh.

Wagner then continues by devising several algorithms for vertex normal computation, improving each one gradually. The first version of the algorithm simply iterates through the list of vertices, calculating the surface normal of each triangle that shares the current vertex, and adding it to a tally, to finally normalise the vertex normal.

His subsequent algorithms solve problems related to lighting, such as surfaces that were not meant to be smooth by adding a threshold value to detect edges, and the different sizes of the triangles sharing the same vertex, for which he adds the triangle's surface as a weighting factor.

Thereafter, Wagner describes improvements that can be done when applying his final algorithm to real-time generation with deformable meshes, such as pre-computation to acquire knowledge of which triangles share which vertex, and surface approximation if accuracy can be sacrificed in exchange for a faster alternative. Finally, if the surface is defined with a regular grid of vertices, calculating the polygon normals in one go, and only then using the original algorithm without an adjacency list, using indexes of a given vertex to generate the neighbouring faces.

This paper was decisive when designing our vertex normal computation algorithm. The first algorithm described was used modifications to suit our project. We discuss our algorithm in section III, part C of this paper.

Another paper, written by Jirka and Skala (2002), contains a description and comparison of five methods for triangle mesh vertex normal computation. A comparable paper was

written by Jin S., et al. (2005) but will not be discussed here. We see that the first algorithm proposed by Wagner is equivalent to the Gouraud method, who suggests computing the average of the normals of the faces neighbouring the vertex. Jirka and Skala (2002) then describe the Thürmer and Wüthrich method that points out the dependency of the normal on the meshing structure. To solve this issue the method uses "the size of the angle of the facet's edges incident to the computed vertex" as weights in the calculation. As is correctly observed, both methods smooth out every single edge, which is not necessarily the desired effect, especially when lighting scenes where sharp edges are features of the scene. They suggest defining a "decision angle" (p.14, section 2.2 of the *Vertex Normal Computation* chapter) which is exactly what Wagner suggests with the "threshold" value. The third method described has, once again, similarities with one of Wagner's algorithms in that it considers the area of the vertices' surrounding facets for the computation. Finally, the last two methods respectively use linear regression and the finite difference method, and as opposed to the first three methods are not restricted to triangular meshes, but are easily transformable to other kinds of data.

After running tests on each of the five methods, with accuracy as a primary focus, Jirka and Skala come to the conclusion that the Thürmer and Wüthrich method is the best solution. They explain that the linear regression and finite difference methods need to do more complex matrix operations, which explains their weaker performance at the speed test. They also remark that the facet normal methods (the first three) have very similar time performances. Because we mainly focus on the speed for our project, we will choose the simplest method, which is the Gouraud method.

It is interesting to note the great similarities between concepts behind the algorithms presented in Wagner's paper and the methods compared in Jirka and Skala's paper. Although Wagner's is not a published paper, but technically just a "tutorial" that we assume was written without much research for related work in the world of scientific papers, we observe that methods can be elaborated seemingly outside the "official" research world, that are very similar to "official" methods associated with researchers' names. We like to find surprising similarities in works that come from two different "worlds".

B. Polygon Mesh Data Structure

In his dissertation paper, Colin Smith (2006, pp.129-137) discusses the advantages and drawbacks of existing data structures for polygon meshes. The main focus of his thesis was on the VV (Vertex-Vertex) system as a modelling system for geometric and biological modelling. VV in particular is designed to handle dynamical structures, which is ideal for modelling. However, the choice of data structure needs to be done when looking at the application, the type of operations to be performed, the performance need, and the size of the data. Certain data structures will be better for rendering, or topological information access, therefore the requirements will govern the final choice.

VV meshes are the simplest representation of a mesh structure being a set of vertices connected to other vertices. Smith (2006, p.130) states that "the simpler the data structure, the easier it is for a user to understand and figure out how to apply transformations" and that the amount of memory required when storing an object such as a polygon is less than that required by other data structures when using VV. The face and edge information is implicit which does not facilitate operations on those elements of the mesh.

The most widely used mesh representation is the face-vertex mesh structure due to its

simplicity and is used in the OBJ file format, a universally accepted open file format that can be imported/exported by the major 3D graphics applications (such as Blender, 3D Studio Max, Autodesk's Maya, and many others). In this case, the structure uses faces that each have pointers to an ordered list of vertices. (see Figure 1.) Each vertex has a list of its surrounding faces, which can vary in length between vertices. Explicit lookup of the adjacent faces to a vertex and of the vertices of a face is made possible. Smith states that it is only possible to traverse from faces to vertices, but not the other way as there is "no concept of neighbouring vertices across faces" which renders this structure inconvenient to use in many situations.

The next polygon mesh structure Smith evaluates is the Winged-Edge structure. He describes it so: "Every face points to its edges and each edge points to two vertices. Each edge also points to its four neighbouring vertices. Each vertex points to the edges it is part of and each edge also points to two pairs consisting of the neighbouring face oriented by the edge on the counter-clockwise rotation of each face". We believe there is a small mistake in this description. Instead of each edge pointing to its four neighbouring *vertices* it should be pointing to its four neighbouring *edges*. We verified this in Baumgart's paper "Winged-Edge Polyhedron Representation for Computer Vision" (Baumgart, 1975) who states that "each edge node contains a link to each of its four immediate neighboring edges clockwise and counterclockwise about its face perimeter as seen from the exterior side of the surface of the polyhedron". Notwithstanding, Smith writes that the data structure is advantageous in that one can get from any element to any other in a straightforward way, but that the primary drawback is that it requires a large amount of storage because of all the pointers required. This also increases complexity when making modifications.

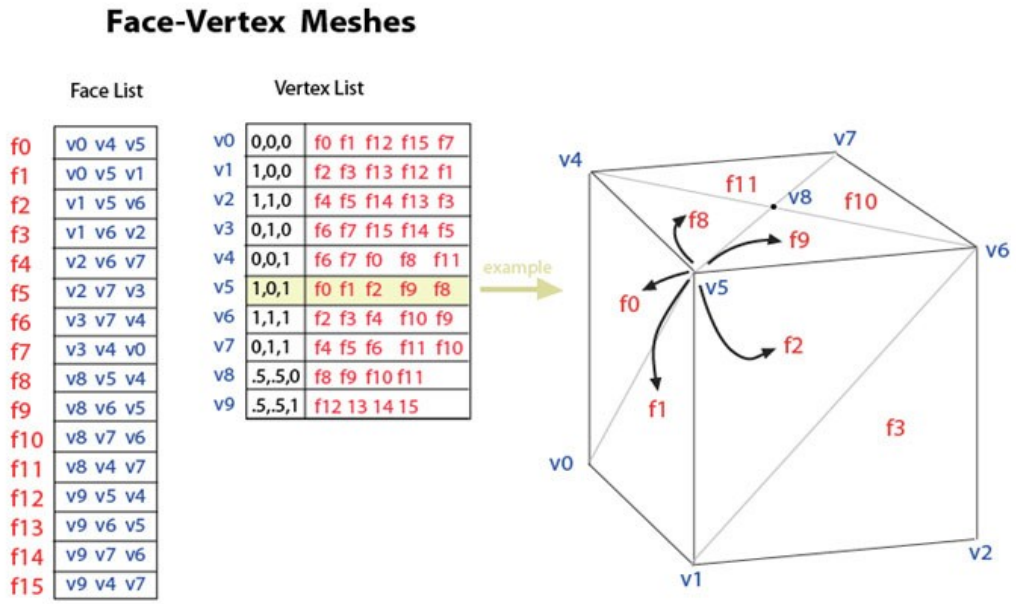


Figure 1: Face-Vertex mesh structure. (Rchoetzlein, 2008)

Half-Edge meshes follow, which are similar to Winged-Edges meshes to the exception that they use half of the edge traversal information (half-edges), in effect using vertices, half-edges, edges and faces (sometimes without them).

The Quad-Edge structure does not use faces in its definition, unlike the previous data

structures Smith presented, only using the connectivity between edges and half-edges, and vertex information.

Finally, Smith presents the Corner-Table data structure which he explains was created to store triangle meshes as triangle strips in a compact way. Each vertex is stored in an array and has the indices of its neighbour vertices stored in more arrays. Local structure of the mesh is easily allowed by the structure, and the corner-tables can be compared to the triangle fans used in graphics, which is a very efficient way of representing polygons.

This thesis is a useful paper when trying to design the most suitable mesh data structure, which we were required to do quite soon in the project. Yet, it was only discovered much later, near the end of the project, which means that it could not be used effectively. It is interesting to see how our mesh data structure, which we designed in an incremental and evolutionary manner, compares to those mentioned above. The project's mesh data structure is described in the next section.

III. Solution

In this section, we discuss the design and implementation of the solution. Firstly, we look at the architecture and design of the solution, then the implementation is discussed, looking specifically at the algorithms used as well as some of the problems encountered.

A. Architecture and Design

Before starting to design and build the solution, the tools had to be selected. First of all, the programming language, which is one of the main factors that affects speed: C++ was chosen because of its speed, robustness, and portability across platforms. We had previous experience of the C/C++ languages which also influenced our choice. Similarly, we had experience with OpenGL as graphics visualisation tool. OpenGL is a cross-platform API (Application Programming Interface) which is widely used in industrial applications to display 3D images from simple geometric shapes and easily callable with C/C++. More specifically, the GLFW framework was chosen, once again because we had previous experience of that framework. The development platforms that were used are the Windows XP and Windows Vista Operating Systems (OS) because of our familiarity with them, and the widespread availability across the university of XP.

A number of different design models are available for software development. All have pros and cons that should be considered before adopting any. Because the requirements were likely to change as familiarity in the project domain grew, certain models that are too inflexible were rejected, such as the waterfall model whose stage partitioning is too rigid and makes it difficult to respond to changes in the requirements.

The evolutionary iterative development process, in combination with evolutionary prototyping, is the model that fitted this project quite naturally. Larman (2003, p. 15) explains in his book on agile development that “Evolutionary iterative development implies that the requirements, plan, estimates, and solution evolve or are refined over the course of the iterations, rather than fully defined and “frozen” in a major up-front specification effort before the development iterations begin.”. Being the only person working on the project and reporting to our supervisor, this development structure was ideal, allowing us to continually add new features and optimise the programme, as well as getting feedback from our supervisor on the prototypes.

The architectural pattern we used for the software processing model is similar to the graphics pipeline in that it deals with the input, and outputs the processed information to the screen. An addition to this simple pipeline model is user input to modify what is being displayed. We created a loop between the graphical output and the user input stages which means that the images displayed can be modified by the user. Note that the data input and processing stages only deal with the point cloud and mesh data. The only changes the user can make is to the viewing options, such as camera position; no modifications to the point cloud data can be made. The reasoning behind this choice of processing model is that all the calculations (normals, gradient variation, colours...) are done initially and the results are stored for later use when rendering. If that information had to be calculated every time the user changed some graphical output option, a lot of redundant calculations would occur, creating extra overhead and processing time, and affecting performance significantly. A drawback of the chosen model is that all that preprocessed information has to be stored in memory, and the memory required therefore greatly increases. Nonetheless, memory being more cheaply available than raw processing power nowadays, using more memory versus poor performance was an easy choice.

In order to do any sort of calculation, extra information about the points needs to be known, such as point (or vertex) connectivity. From an initial set of points, one can apply triangulation algorithms to establish a structure to the point cloud, and information about possible vertex connectivity. This step was outside of the scope of the project, and will be discussed shortly in our conclusion. We decided to create a simple mesh structure that we could easily store and read in using files. By keeping the mesh rectangular with even spacing between each vertex, and only changing its height, width, and distance between vertices to modify its size, the structure would be easy to make. A basic programme was written to enable fast creation of those mesh files, and allowing the user to specify the size using the three parameters mentioned above. Note that the created mesh is a simple grid of points, evenly spaced out and at the same depth: in an 3D *xyz* coordinate system, the points all have the same *z* (i.e. depth) value. Tables 1 and 2 show samples of the mesh files. We kept the vertex information separate from the triangle information, which makes it clear which files contain which information. The main drawback is that two files are required in order to display the mesh. For a project this size, this does not necessarily make a huge difference, but storing all the information in one file would be the best solution in the long term and would not require too much code change. Our efforts were not concentrated on this aspect however.

Another issue was to build meshes with some geometry. Indeed the mesh creation programme placed all the vertices on the same plane, which makes a flat mesh. This is not interesting for our project since we want to study the surface changes. Unfortunately, we didn't have time to implement some tool to make a mesh with variable geometry, or to manipulate the meshes. All the changes had to be done by hand in the vertex files. This is obviously not a sustainable method.

There is a specific vertex number order we imposed when creating the triangle file. Indeed, as we will see in the implementation section, when creating the triangles, in order to have all surface normals pointing the same way, the vertices had to be added in clockwise or anti-clockwise order consistently. We chose anti-clockwise.

Table 1. verticesXXxXX.txt sample file. Each line contains the x, y and z coordinates of a vertex. The first line is vertex number 1, the second line vertex number 2, and so on. (The capital Xs in the file name are replaced by the width and height of the mesh)

```
-3.50000,3.75000,-1.00000,
-3.00000,3.75000,-1.00000,
-2.50000,3.75000,-1.00000,
-2.00000,3.75000,-1.00000,
...
```

Table 2. trianglesXXxXX.txt sample file. Each line corresponds to a triangle and each number to a vertex. For instance the first triangle consists of vertices 1, 16, and 2. (The capital Xs in the file name are replaced by the width and height of the mesh)

```
1,16,2,
2,16,17,
2,17,3,
...
```

Our primary objective required us to display a polygonal mesh. There are different types of elements that can be stored and used to display a polygonal mesh as discussed in the related work section: vertices, edges, faces, polygons, and surfaces. A vertex is a position with which we often associate other information such as colour, and a normal vector. An edge is simply a connection between two vertices. A face is constituted of three edges at least since it has to be a closed set. A set of faces makes a polygon. Surfaces group smooth parts of a mesh.

Not all of these elements need to be stored in the mesh data structure in order to display or manipulate it. We decided to use vertices, edges, and faces (i.e. triangles). The final structure is detailed in Figure 2.

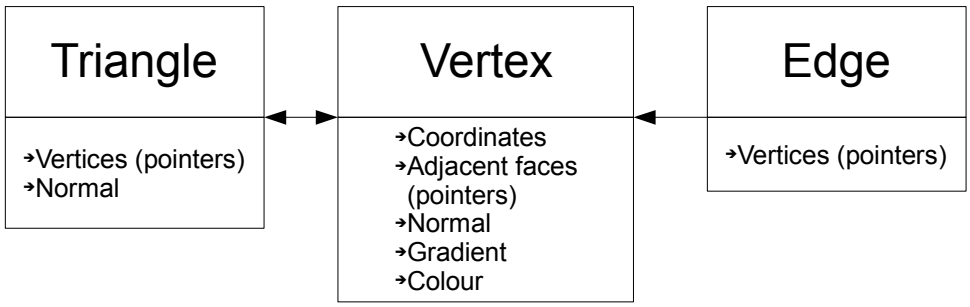


Figure 2: Mesh Data Structure used for the project. Arrows indicate one or more pointers to the box pointed to.

In hindsight, and after discovering Smith's (2006) paper, we can observe that our mesh model is largely similar to a Face-Vertex mesh structure, with the addition of edges. The edges are not essential to the project, and can be implicitly defined. In future development, removing them would be a likely decision.

B. Implementation

Initially, a simple application was built to read and display point cloud data files, which

helped gain more familiarity with C++ and OpenGL. The sample file provided by our supervisor contained 186'000 points, and a segmentation error occurred as we tried to display all 186'000 points when using the Windows XP OS. Interestingly, that error did not occur when running the same programme on Vista. We suspected a memory management problem, that could also be related to the way the OS manages the memory. The problem was solved by changing the code and using dynamic memory allocation.

This first prototype was only an initial file reading and graphical display test. Our main aim being to calculate, or estimate, the gradient variation at each vertex. To attain this objective, we decided to first write a new prototype that would make a very simple mesh and calculate the normal for each of the triangles, then build the additional features – vertex normal estimation, and gradient average calculation – onto that simple prototype, therefore following the evolutionary prototyping method we chose. Vectors are required when calculating the normals, therefore, after a bit of research, a C/C++ header file containing a vector class and vector functions written by Bryan Duff was included in the code. This enabled us to write the algorithms required by the project, which are described in section C.

Initially, our data structure only consisted of triangles (faces) and vertices, of which we kept a list. Each triangle kept a pointer to its three vertices, but no information about surrounding faces was kept in the vertex class. We then started storing edge information in order to be able to draw edges without any hassle. Little by little, as needs increased, more and more information was stored, and the mesh data structure grew accordingly, when it could not cope with the demand.

Normalised vertex normals were then added in the rendering. This highlighted the fact that all normals were not pointing the same way when supposed to. The reason for this is that when doing the cross product of two edges of the triangle, we get a normal pointing the opposite direction if we swap the order we pick the vertices: Figures 3.(a) and 1.(b) show an example of a triangle with the vertices set in a certain order. If the vertex labels v2 and v3 were swapped over, vectors e1 and e2 would also be swapped over (assuming we keep using their respective vertex labels), which would result in the normal N pointing the opposite direction. This small problem was very useful to highlight an important fact to consider when later creating the mesh data structure, algorithms, and the mesh data file creator and reader: Triangles had to keep an ordered list of their vertices. In computer graphics, the standard convention for the ordering the vertices is anti-clockwise, which we adopted.

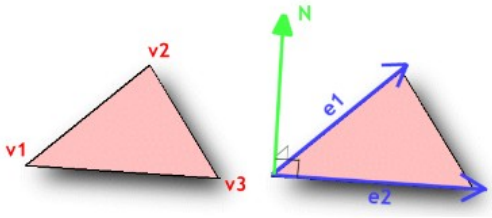


Figure 3. Triangle surface normal calculation using the cross product of vectors e1 and e2. Figure 3.(a) is the triangle on the left with vertex labels, and Figure 3.(b) is the rightmost one with the vectors.

We attempted to keep our programme modular, and keep related functionality in the same files. Figure 4 gives an overview of the structure of our programme. It forms a tree structure,

with the *main.cpp* file being the root of the tree, and controlling the whole application. Because *main.cpp* includes the *mesh.cpp* header file which itself includes the *struc.h*, *PhysicsMath.h* and *filehandler.h* header files, *main.cpp* can actually access all the public functions in those files.

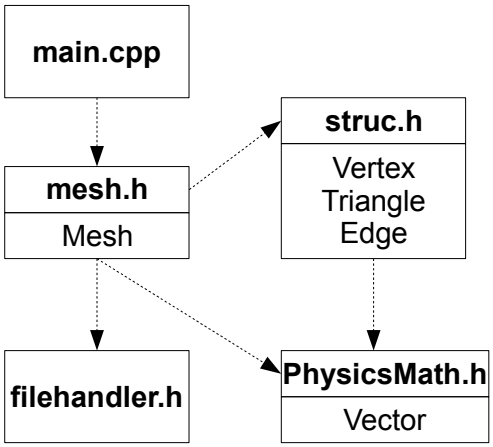


Figure 4. Programme structure. An arrows indicates use of C/C++ preprocessor command "#include".

To test our algorithms we had to measure the time they took to process the data. The method we chose is quickly described in section IV. One of the issues we did have when choosing the timing method was the accuracy of the timer. The first method we tried made use of the *time.h* library which has a *clock()* function which returns the number of clock ticks since the computer was started. We simply need to subtract the number of clock ticks done at the end of the computation from the number done before, and divide by the C constant *CLOCKS_PER_SEC* to get the number of seconds between both calls to *clock()*. We can even get the result in milliseconds by multiplying the result by 1000. However, it was not more accurate that that, and for small inputs, our algorithms were too fast and all returned 0ms. We then found that the *windows.h* had a *QueryPerfomanceCounter()* which could be as precise as nanoseconds, which was what we needed.

C. Algorithmic

In this part, we will describe the algorithms used in our solution. They consist of the triangle normal, vertex normal, and gradient variation calculation algorithms.

Firstly, triangle normals had to be calculated. A simple function was added to the Triangle class in the programme (see Algorithm 1). The code used calculates the normal vector of a plane. It was written by GGB and was found on Microsoft's support website (GGB, 1995). We modified it to make it usable with the code that already existed.

Algorithm 1. C++ code for the calculation of a surface vector normal.

```
void Triangle::ComputeNormal()
{
    float Qx, Qy, Qz, Px, Py, Pz;
    Qx = vertex[1]->x-vertex[0]->x;
    Qy = vertex[1]->y-vertex[0]->y;
    Qz = vertex[1]->z-vertex[0]->z;
    Px = vertex[2]->x-vertex[0]->x;
```

```
        Py = vertex[2]->y-vertex[0]->y;
        Pz = vertex[2]->z-vertex[0]->z;

        float fNormalX = Pz*Qy - Py*Qz;
        float fNormalY = Px*Qz - Pz*Qx;
        float fNormalZ = Py*Qx - Px*Qy;

        normal = Vector( fNormalX, fNormalY, fNormalZ );
        normal.Normalize();
    }
```

The second algorithm we needed was the vertex normal algorithm. Once all the triangle normals are calculated, we can estimate the vertex normal by averaging the normals of the faces adjacent to the vertex. In effect, we simply add the vectors together and normalise the total, which has the same effect. Algorithm 2 shows the pseudocode for this. As seen in the related Work section in subsection A with Wagner's (2004) and Jirka and Skala's (2002) papers, several methods are available for vertex normal calculation. For our particular needs, a quick estimation method was sufficient.

Algorithm 2. Pseudocode for vertex normal estimation.

```
struct Vertex( position p, Normal n, List adjacentFaces )
struct Face( Normal n )
for each face in v.adjacentFaces
      n = n + face.n
endfor
n = Normalize(n)
```

Third and last, our gradient variation estimation algorithm. We now have normal vector estimates for all the vertices in the mesh thanks to the first two algorithms. We can measure the angular relationship between two vectors using the dot product.

Let A and B be vectors. The values $|A|$ and $|B|$ represent the lengths of vectors A and B, respectively, and Θ is the angle between the two vectors. The following relationship can be defined:

$$A \cdot B = |A| * |B| * \cos(\Theta) \quad (1)$$

Because all normals are normalised (set to a length of 1) we can rewrite Eq. (1) as:

$$A \cdot B = \cos(\Theta) \quad (2)$$

and from which we get:

$$\Theta = \text{acos}(A \cdot B) \quad (3)$$

Thus, to measure the angle between two vectors, we can compute the dot product between those two vectors. If the two vectors are perpendicular, the result of the dot product will be zero; if it is less than 90 degrees, the dot product will be greater than zero; and if it is greater than 90 degrees, the dot product will produce a negative number (properties of the cosine function).

We then retrieve the angle with the arc cosine function (Eq. (3)) and convert the result from radians into degrees for an easier readability of the values when testing.

Running the algorithm on test cases, we noticed some results were not what was expected. For instance, when calculating the dot product of normal vectors of a flat mesh, the expected result for all calculations was zero degrees, since they should all be parallel, but a few calculations returned some different result. This is due to precision when doing calculations, which is never perfect. To resolve this issue, we set a tolerance value that serves

as a threshold to make sure dot product results very close to 1.0 are set to 1.0, and therefore that the correct value is returned for *flat* surfaces.

The dot product is computed between each vertex and its neighbour vertices in turn. To start with, we only averaged the angular results returned by the dot product. Later on in the project, we decided to store the maximum result as well as the average in order to test what difference it made when visualising the results on the OpenGL rendition of the mesh. Algorithm 3 shows the pseudocode for the average and maximum gradient calculation.

(Note: Our dot product function was implemented by Bryan Duff in his header file we included in the project; see the previous subsection)

Algorithm 3. Pseudocode for average and maximum gradient calculation.

```
struct Vertex( position p, Normal n, List adjacentFaces,
               List neighbourVertices, Gradient avgGrad,
               Gradient maxGrad )
VertexList v
dotTolerance = 0.000001

function CalcGradient( Vertex v, Vertex v2)
   dot = dot_product(v.n, v2.n)
   if( absolute (1.0 - dot) < dotTolerance )
       dot = 1.0
   endif
   result = ( acos(dot)*(180/PI) )
   return result
endfunction

function CalcAvgGrad ()
   for each v in VertexList
      Counter c = 0
      for each v2 in v.neighbourVertices
         tmp = CalcGradient( v, v2 )
         v.avgGrad  = v.avgGrad + tmp
         if( maxGrad < tmp )
            maxGrad = tmp
         endif
         c = c + 1
      endfor
      v.avgGrad = v.avgGrad/c
   endfor
endfunction
```

D. Visualisation

One of the aims of this project was to display the point cloud in a mesh structure and highlight the areas of important surface variation. At first, the mesh was displayed using OpenGL by drawing the wire frame (only the edges of the triangles) but this clearly led to confusion as to the geometry of the mesh when triangles were overlapping each other. The triangles were filled with an arbitrary colour, which made the shape more visible. However, a problem appeared: the depth value did not seem to be taken into account when drawing the objects, which meant that although a triangle was in front of another triangle in 3D space, if that “hidden” triangle was drawn more recently, meaning if it was created in the code after

the first triangle, then it would appear to be in front of that first triangle. The simple solution was to enable the OpenGL depth testing, one line of code which we had somehow omitted to write when creating the programme. This is an example of a problem with a (stupidly) simple solution that can cause a lot of trouble.

Initially, rendering the mesh with OpenGL proved to be slow, much slower than expected from a specialised graphics API. Displaying 900 points is supposed to be a "piece of cake" one could say, especially when considering the reasonably powerful (not high-end) dedicated graphics card used in the testing configuration, but the frame rate was sluggish (5 to 20 Frames Per Second (FPS)). The problem had to come from somewhere else, and we soon worked out that we were iterating through the whole vertex and triangle lists every single frame OpenGL rendered. We first tried to improve our data structure by using different containers. This proved successful to some extent. For instance, our Vertex class stored a list of adjacent faces in a C++ standard library *list* container. As soon as we changed from a list to a standard library *vector* container, performance increased significantly. We also noticed, with the *vector* container, that using a C++ iterator, rather than directly accessing the data with indices, increased performance from 250 to 320+ FPS on a mesh with 3200 vertices.

Still, making small improvements in the data structure did not solve the real problem, which was the constant iteration through the list of elements to display. Some research lead to the use of OpenGL display lists. A display list is a group of OpenGL commands that are compiled together and stored for later execution. This is one of the fastest methods to draw static data because all the commands and data are cached in the display list, which reduces the number of processor cycles to do. Nonetheless, they have a disadvantage: A display list cannot be modified once it is compiled, and that list also needs to be stored in memory. Therefore if a large number of instructions are stored, an equally large amount of memory is required to store the list.

Highlighting the surface changes was one of the aims of the project. Once the average and maximum gradients were computed, we interpolated the colour we would draw each vertex in according to the magnitude of the gradient. Colours in OpenGL are expressed as values between 0 and 1. We therefore decided to divide the gradient value by an arbitrary number or 70. We estimated, after much informal testing on meshes that not many surfaces would not have a gradient variation of more than 70 to 90 degrees, because extreme changes are not easily recorded by laser range scanners. Surfaces hidden by other surfaces would not appear in the resulting point cloud data, and the high surface variation would not necessarily be reconstructed when triangulating the point cloud.

Navigation in the three dimensional scene where the mesh was being drawn was implemented in order to facilitate understanding of the structure displayed. We decided to enable different display modes: vertices only, edges only, surface only, surface with edges (to see the triangles), and vertex normals. Movement in the 3D space was developed which helped us view certain portions of the mesh by zooming in, or zoom out to see the whole structure. The mesh can be rotated in all directions, and the camera can be moved along a 2D plane (up, down, left, right).

IV. Results

In this section we present the results of our tests, focused on the time the algorithms took. All the tests were run on a Dell Inspiron 6400 laptop, with an Intel Core Duo T2250 processor

clocked at 1.73GHz, 2 GB of DDR2 RAM, ATI Mobility Radeon X1400 graphics with 128 MB video memory, and Windows Vista 32 bit Business edition.

In order to measure each algorithm's running time we decided to use a Windows specific timing function, found in the windows.h C header file. It contains a function called *QueryPerformanceCounter()* that provides high resolution timing (down to nanoseconds). This makes the programme dependant on Windows, but this was decided because all the tests would be run on a Windows Vista OS. We run the algorithms on each input size at least four times, and for smaller inputs we had time to run the algorithm 20 times on the same input. We believe this number of iterations is sufficient to capture the global trend of the computational time requirements. The results were quite consistent which did not push us to run more tests.

A. Speed Results

Figure 5 is a scatter graph that shows the total time it takes to calculate the normals for both the triangles and the vertices. Figure 6 is a detail of the first graph showing the smaller inputs which are not clear on the first graph. As we can see, the results follow a linear trend. The small inconsistencies around the trend line can be explained by extreme values returned by calculations done in slightly different testing conditions.

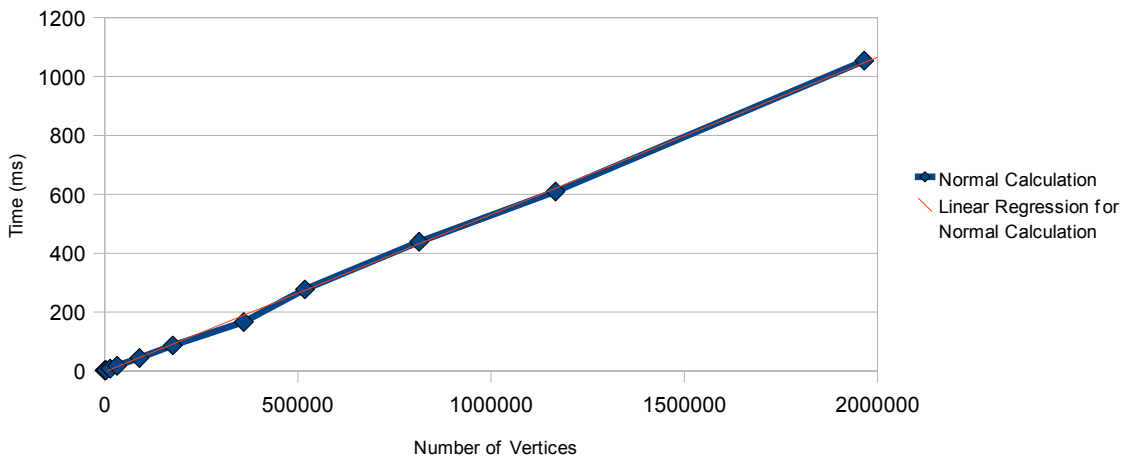


Figure 5. Scatter Graph with trend line (linear regression) of the total surface and vertex normals computation time.

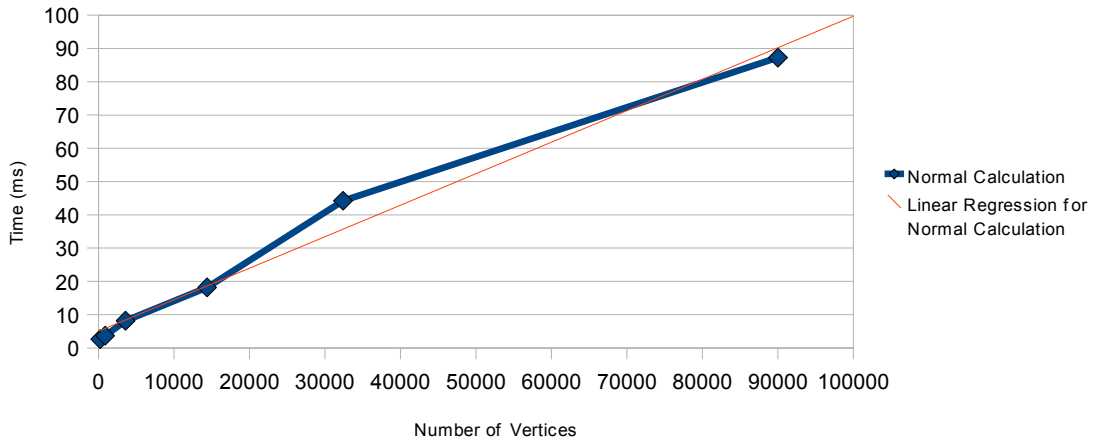


Figure 6. Scatter Graph with trend line (linear regression) of the total surface and vertex normals computation time. (detail)

Figures 7 shows the total time the gradient calculation took, and Figure 8 a portion with the smaller values to see more of the detail.

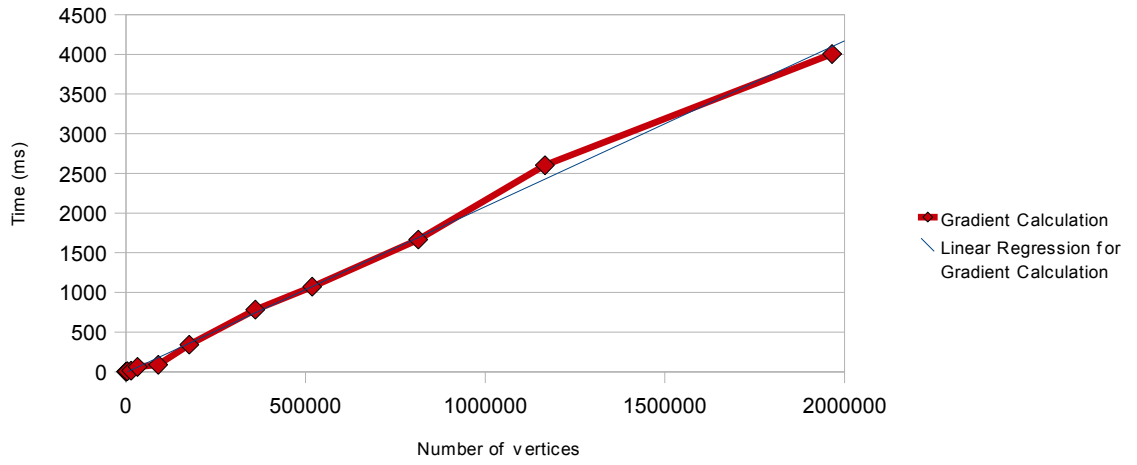


Figure 7. Scatter Graph with trend line (linear regression) of the total gradient calculation time.

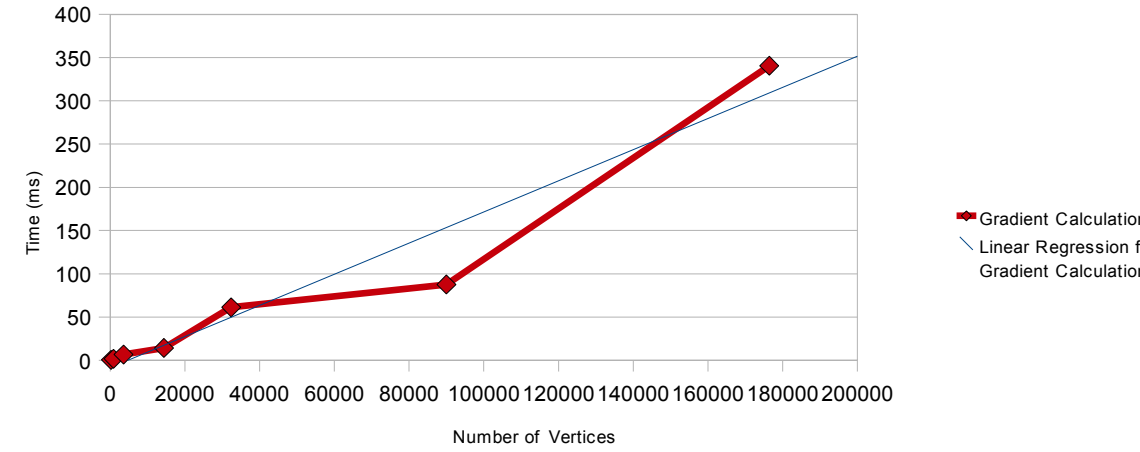


Figure 8: Scatter Graph with trend line (linear regression) of the total gradient calculation time. (detail)

B. Rendering

No extended testing was done on the visualisation part of our programme because our focus was set on the algorithmic part. We display the number of frames per second shown dynamically in the title bar of the OpenGL window. This enabled us to detect the changes any modification to the code made to the rendering speed. The main information we could extract was that OpenGL display lists increased the number of FPS by a considerable amount. If using a display list, we observe a 400% performance increase from 4 FPS to 12 FPS when showing a mesh of 813604 vertices with edges and the surface being displayed. Again, if displaying 90000 vertices, an increase from 5 to 50 FPS can be noted, with the same elements displayed.

C. Other Observations

From table 3 we can see that the file reading process is the most time-consuming element of the programme (excluding displaying). The file sizes for the biggest mesh we tested, with 1965604 vertices, is 91 MB for the triangle connectivity file, and 52.5 MB for the vertex coordinates file. The file reader therefore needs to process about 134 megabytes of data before the actual computations can take place.

Table 3. Average total times for file reading and the sum of all operations (file reading, normals and gradients) in milliseconds.

Averages		
Number of Vertices	**Total Time**	**File Reading**
225	23.3	20.99
900	75.0	72.40
3600	285.8	281.79
14400	1128.2	1115.01
32400	2245.1	2214.98
90000	7095.9	7124.89
176400	12811.9	12665.65
360000	27995.5	27627.53
518400	40366.4	39898.76
813604	62875.0	62145.45
1166400	91676.0	90462.25
1965604	155425.8	153759.50

No particular tests were run to measure memory usage. However, we observed a memory usage peak of 1.4 GB when running the tests on the biggest mesh (1402 by 1402 vertices, 1965604 total). This clearly shows memory management was not our main concern and further work on the data structure can be done to optimise the solution in this area.

V. Evaluation

A. Algorithms

In order to evaluate the algorithms and their implementation, we decided to estimate the time complexity of the algorithms so that we could compare it to the actual time complexity when running them in our programme. The details for the theoretical computational complexity analysis will not be given here, but they were done using algorithms 1, 2 and 3 described in section III and a running time complexity estimation of the C++ functions used. We do not explain these mathematical terms here and assume the reader has a basic knowledge of computational complexity.

First of all, the surface and vertex normals calculation algorithms both work in the same way by iterating through all the triangles/vertices and running their respective algorithms, Algorithm 1 for triangles and Algorithm 2 for vertices, for each of the list elements. Iteration through a list takes time $n+1$ on a list of n elements, which is $O(n)$. We need to know what the complexity of the operations run in the iteration loop is.

The triangle normal computation is very straightforward and simply takes constant time $O(1)$. As for the vertex normal computation, it is slightly more complex in that for each

vertex we have a nested loop to add the normal values for each adjacent vertex. However, because our mesh has a predefined structure – regular triangle mesh – we can cap the number of neighbours each vertex has to a maximum of six. In the end, the vertex normal calculation function also takes constant time to run, which brings both triangle and vertex normal algorithms to a running time of *O(n)*.

Secondly, the gradient computation is done in two separate functions as can be seen in Algorithm 3. The gradient calculation between two single vertices is done in constant time, *O(1)*, and is called by the average and maximum gradient estimation function for all neighbouring vertices of the vertex being processed. Once again, because of our mesh structure we know this will be done a maximum of six times, which is equivalent to a constant time. Finally, this is done *n* times, *n* being the number of vertices in the mesh, which leads to a total running time of *O(n)* for the gradient estimation algorithm.

When comparing to the recorded times when running the algorithms in our implemented solution, we observe that the trend lines are linear, which fits perfectly with our theoretical analysis of the algorithms.

These algorithms are far from being optimal, as redundancy can be spotted in Algorithm 3's *CalcGradient* method. Indeed, this function calculates the angular relation between two vertex normal, but the result is only stored in the vertex that is being processed. By storing the result in both vertices at the time of calculation, computation time can be saved and the algorithm could therefore be faster.

When researching new algorithms, one ideally wants the smallest time complexity possible. Although our solution runs in linear time, it is not extremely efficient when dealing with big inputs.

B. Data Structure

Our implementation of a data structure had both advantages and pitfalls. Because we created it and added to it depending on what our particular needs were at the time, not enough thought was put into the memory space it would require, which is rather critical when using huge datasets. This partially explains the large amount of memory it requires simply to do the calculations on the bigger meshes (see subsection C of section IV for details).

Storing too many variables, too much information, in the data structure means it is not straightforward to use, but has a lot of functionality. However, this functionality is at the expense of efficiency in memory use.

We also note that introducing display lists and therefore avoiding repetitive iteration through the lists of elements in the mesh greatly improves the performance of the programme. This highlights the fact that the data structure used is a bottleneck for the OpenGL part of our solution. Further research into how to create a data structure that can speed rendering would be an excellent direction to follow for future study.

C. Rendering

Another partial explanation for the memory use issue mentioned in the subsection above can be explained by the use of display lists. As explained previously, the more data and OpenGL commands stored in the display list, the more memory is required. With huge datasets, this explains the impact on memory. Indeed we used a display list for the triangles, the edges and the vertices. More control over what operations are done to display the elements would enable a better memory usage.

A good visual experience is always appreciated when using anything graphical. Just like in video games, whenever the frame rate drops below a certain point, the user experience suffers massively. We attempted to optimise the rendering part of this project with the aim that future work on this project might lead to the creation of a usable tool to visualise areas that could be of interest, especially in the geological field of study, where analysing rock outcrop laser scans can take a considerable amount of time. Even so, this was not the primary objective of this project, which is why it is not fully optimised.

D. Testing, Version Control, Backup and Organisation

A modular approach was adopted by choosing the evolutionary development process. We combined this with a simple version control system that consisted of creating a new folder named following this convention: *NameYY.MM.DD* with *YY* being the year, *MM* the month, and *DD* the day of the creation, e.g. *Mesh09.02.15*.

Consequently, when a new feature or some change in code was applied to the main prototype, if a problem was discovered when running the programme on tested inputs, debugging could take place, and in the worst-case scenario, a roll-back to the most recent working version could be done easily. The version control doubled as a backup system by saving the different versions on external storage in case of computer problems that could lead to data loss.

The project plan was designed at the start of the project, with a naïve view on how difficult or easy certain tasks would be. Although initial soft deadlines were set, they were often modified or new ones set weekly during the meetings with our supervisor. The hard deadlines were useful to make sure we knew when to start a new development stage.

VI. Conclusion

The main aims of this project were to design and implement an application that would enable the visualisation of a point cloud dataset displayed as a mesh and highlighting of the areas with important surface variations, followed by an analysis of the algorithms created. We implemented a mesh data structure and developed algorithms that would calculate the average and maximum gradient variation, following our definition of gradient. OpenGL was used to display the mesh and visualise the areas of interest. We noted that the theoretical analysis of the computational complexity of our algorithms was in accordance with the speed tests run on the implemented application, which shows our solution to run in linear time. The graphical rendering part of the project encountered many issues during the implementation stage, and can still be optimised, but most of the tested sets of data could be comfortably visualised. The problems helped us better understand the mechanics of graphics and how they work with data structures.

Many improvements can be made to the current solution. The mesh data structure can be simplified, and small short cuts can be employed in the algorithms. Ideally, an algorithm with a running time of the order of *log* would be possible. This will probably involve use of different data structures to store the mesh elements, and making use of multi-threading now that multi-core CPUs are becoming more and more widespread in consumer computers.

The utilisation of different containers that seem quite similar but improve of decrease the performance of certain operations was interesting. A deeper analysis of the limitations and advantages of the different C++ standard library containers could be useful as a side-project.

The algorithms we wrote only make use of each vertex's direct neighbours, but a good

aim for further work would be to extend this to include a larger area of the mesh and test the different possible ways to include them (How far to go? What weight to give the further vertices? How to store them and access them?).

We feel this solution can be used to create a tool that will highlight interesting areas in rock laser range scan, but we would have to undertake some dialogue with geologists in order to set the requirements and the different ways to identify features in outcrops.

Originally, the idea was to use raw data from laser range scans. However, as previously stated in this paper, the raw data does not contain any information on connectivity, and certain operations such as triangulation are necessary before we can run any algorithms similar to the ones developed in our solution. The creation, either of a triangulation tool to apply a method such as Delauney 2D triangulation to the data, or of a small programme to enable input of pre-computed files using standard file formats such as VTK (Visual Toolkit) would enable a better integration of the resulting programme in the graphical world of computing as well as a wide access to real-life data. More thorough testing using data from the real world would help define new requirements for improvement.

References

Baumgart B., (1975). “Winged-Edge Polyhedron Representation for Computer Vision”. *National Computer Conference.* Stanford, California. Available at: http://www.baumgart.org/winged-edge/winged-edge.html (Accessed: 02 May 2009)

GGB, 1995. *How to Set the Current Normal Vector in an OpenGL Application.* [Online] (updated November 2006) Available at: http://support.microsoft.com/kb/131130/en-us (Accessed: 02 May 2009)

Jin S., Lewis RR., and West D., (2005). “A Comparison of Algorithms for Vertex Normal Computation” *The Visual Computer*, School of Electrical Engineering and Computer Science, Washington State University. Springer.

Jirka T., Skala V., (2002). “Gradient vector estimation and vertex normal computation”, *Technical Report No. DCSE/TR-2002-08*, University of West Bohemia in Pilsen.

Jones, R.R., McCaffrey, K.J.W., Clegg, P., Wilson, R.W., Holliman, N.S., Holdsworth, R.E., Imber, J., and Waggott, S., (2007). “Integration of regional to outcrop digital data: 3D visualisation of multi-scale geological models.”, *Computers & Geosciences*, v.33.

Kurz T.H., Buckley S.J., Howell J.A., and Schneider D., (2008). “Geological outcrop modelling and interpretation using ground based hyperspectral and laser scanning data fusion”, *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Available at: www.tu-dresden.de/ipf/photo/publikationen/2008/Kurz_ISPRS2008.pdf (Accessed: 03 May 2009)

Larman C., (2003). *Agile and Iterative Development: A Manager's Guide.* Addison-Wesley Professional.

Rchoetzlein (wikipedia user), (2008). Face-Vertex Mesh [Image]. Available at: http://en.wikipedia.org/wiki/File:Mesh_fv.jpg (Accessed: 04/05/09)

Smith C., (2006). *On Vertex-Vertex Systems and Their Use in Geometric and Biological Modelling*. Ph. D. dissertation, University of Calgary.

Wagner M., (2004). *Generating vertex Normals*, unpublished. Available at: http://www.emeyex.com/site/tuts/VertexNormals.pdf (Accessed: 02 May 2009)

White P.D., Jones R.R., 2008. “A cost-efficient solution to true color terrestrial laser scanning”, *Geosphere*, v. 4; no. 3; p. 564–575.

Object Clustering for Game Environments

Student Name:

Supervisor l

Submitted as part of the degree of **Computer Science** to the

Board of Examiners in the Department of Computer Science, Durham University.

Abstract –

Context: In distributed environment systems, such as multiplayer online games, many users are interacting with other players. Offering interactive response is a key requirement for such systems. An effective way to achieve this is by minimising the computation required for evaluative actions and responses among interacting objects. This can be done by acquiring a high performance object clustering method, which helps reduce the amount of objects to be processed by considering highly related objects only.

Aims: Results from a recently conducted survey indicate that the future of clustering algorithms lies with the development of hybrid methods (Kotsiantis & Pintelas, 2004). The aim of this work is to develop such a hybrid algorithm that produces excellent clusters at high performance in large distributed online environments, while minimising network communication.

Method: We sample the best current clustering algorithms from every category as defined from a comprehensive review on clustering (Jain, Murty, & P.J.Flynn, 1999). We then attempt to extract the key components which determine each algorithm's success, and extend them into a single high performance hybrid algorithm. Alongside using existing techniques, we introduce the concept of categorising object behaviours into primitive shapes, which hopes to contribute to the clustering field by minimising network overhead. Our method is evaluated and compared with recent related algorithms in a highly responsive simulation tool.

Results: Our new algorithm, Object-Mapped Behaviour-Grid (OMBG), produces near perfect non-Euclidean clusters of arbitrary shapes and sizes in constant time complexity according to some predefined resolution. We show an improvement in the reduction of network overhead by categorising object behaviours into primitive shapes, and we observe success with five example behaviours implemented in a realistic multithreaded and hardware-accelerated environment.

Conclusions: Hybrid algorithms, although in their infancy, produce exceptional results which currently cannot be replicated with any single algorithm from another method. OMBG is targeted at distributed environment systems, and it demonstrates the emerging prosperity of hybrid methods with near optimal performance and quality. We show that it is possible to reduce network communication further than the most recent and successful method, Threshold-based *k*-Means Monitoring (TKM), by representing object behaviours as primitive shapes; however the classification process of how to achieve an optimal mapping between object behaviours and shapes is a new open research topic.

Keywords – Clustering, Hybrid, Large Datasets, Grid, Shapes, Behaviour

I. INTRODUCTION

Object clustering is part of the class of unsupervised learning problems: it is the unsupervised process of classifying similar objects, or data, into groups. Clustering methods have traditionally been applied in fields such as spatial data analysis, pattern recognition and image processing, with the key focus of work being in trying to improve the final cluster quality while maintaining efficiency in terms of time and space complexity. In present times, with the expansion of distributed systems using large dynamic datasets, there is a new demand for efficient clustering methods which don't necessarily focus on the final cluster quality; instead

they focus only on the converging quality and time complexity of the algorithm. Furthering this, a distributed system needs to take into account communication overhead from client-server interaction, which is beyond the capabilities of most recent clustering algorithms.

This work hopes to meet the new demand for an efficient clustering method capable of handling large dynamic datasets in distributed environment systems, while minimising network overhead. We achieve this by evaluating the best current existing algorithms and modularising the techniques adopted by them to solve the clustering problem. We then attempt to find problems with the best techniques and selectively extend components from each method in hope of finding an improved hybrid algorithm. In addition to this strategy, we introduce several new concepts into the clustering field such as object and cluster scope, with primitive shapes defining movement behaviours and locally adaptive thresholds.

A. Common Issues

There are some common issues which repeatedly occur in the field of clustering. One of the most frequent issues is finding a good balance between cluster quality and time complexity. Typically, hierarchical clustering algorithms have poor quadratic time complexity, yet offer a good multilevel quality. Partitional methods are often of linear complexity and do not require storing any abstract data structures; however they traditionally require knowing a fixed value k in advance, which reduces their quality. Furthering this, k partitional clustering can be extended to not require a fixed k, as with quality-threshold clustering, however this requires more processing. Another example where this balance becomes an issue is with the treatment of *noise*. Noise defines anomalies in the cluster data where certain objects are not part of any particular group. Some algorithms, such as k-means clustering (MacQueen, 1967) get affected by noise which reduces the quality of the final clusters. Removing the effect of noise from such algorithms requires additional processing. An example of noise in an online game is where a player may *solo* without being in contact or close proximity with other players.

Some clustering algorithms allow for fuzzy relationships. This is where an object may belong to one or more clusters. Within dynamic gaming environments, it will be important to consider fuzzy relationships so that players can interact with other players outside of their groups. However, a problem associated with fuzzy clustering is that cluster analysis becomes more difficult.

B. Real-Scenario Implementation

This work does not focus on the practical issues that occur when implementing clustering methods in distributed environments. Instead, the focus of this work is with the actual development of the algorithm itself. An example implementation which can use clustering to reduce communication overhead is CyberWalk (Ng, Lau, Si, & Li, 2005). This is a distributed virtual environment which allows users at different geographical locations to share information. It achieves this by adaptively partitioning the environment into multiple regions which can then each be handled by one server. The problem addressed by clustering in the virtual environment is classifying related objects into the same regions so that they can be served by the same servers.

One of the benefits for designing clustering methods, for use in games and distributed applications, is that we can optimise our algorithm for use in multiprocessor environments. However, it is important to understand that in allowing an algorithm to run in parallel against a dynamic environment, there will no longer be tight cluster boundaries (Results: Figure 7b). Instead, boundaries will appear speckled as objects move before the algorithm finishes a cycle. Speckled cluster boundaries make assessing cluster quality more difficult.

C. Contribution Direction

In an online game, we can label each player as a dynamic object within the environment. To cluster these objects by Euclidean distance, we need some information about their location which needs to be sent to the server. The obvious solution is to repeatedly send the object's location to the server; however this approach produces large message complexity, making it unsuitable for distributed environment systems due to the client-server bottleneck.

Recent research shows that objects do not need to repeatedly send client location information (Zhenjie, Yin, Tung, & Papadias, 2008), and that an update from the object is only needed after it leaves a scope as defined by some threshold radius. We predicted that we could further reduce the message overhead by also applying a threshold to the clusters. We also decided that a circular threshold was not necessarily always the optimal shape for categorising movement trends. We predicted that other shapes could be used to categorise player movement behaviour.

D. Deliverables

The research question that we ask is, 'Can we improve distributed clustering methods, and implement a new algorithm, by modularising and extending existing techniques, and considering object and cluster scope?'

To achieve this we set project deliverables categorised into basic, medium and advanced objectives. The basic objective was to specify a simple object clustering method, and then implement the method in a simulation program. The simulation program was to have a simple user interface for users to change some settings such as the number of objects in the virtual environment, and whether the object distribution is even or skewed. The purpose of the basic objective was to create a foundation for future work; our algorithm needed to be incremental and upgradable, and the simulation program needed to be able to compare and process multiple algorithms. We achieved the basic objective with an implementation of k-means in a high performance hardware accelerated multithreaded simulation environment, programmed using OpenGL and C++. This tool has the ability to render millions of concurrent dynamic objects and paint them in any distribution scenario using an airbrush. The airbrush has the additional capabilities of being able to change object behaviour between one of six different types, and change the size and density of the paint area.

The intermediate objectives were to enhance the clustering method to handle both non-Euclidean and Euclidean metrics. Non-Euclidean metrics are important in game environments as they take into account players who are interacting with distant players, such as friends communicating via a messaging system. Our k-means foundation needed to be extended to handle non-Euclidean metrics. Unfortunately, the very nature of k-means is to cluster based on the distance to a centroid; which only produces sphere-shaped clusterings. However, OMBG achieves the intermediate objective by not clustering based on distance, instead it groups data by a *winner-take-all* principle between an object relationship function. With this approach, it is possible to alter the weightings in the relationship function such that objects form groups with nearby objects of the same behaviour more frequently than with objects of different behaviours. We also show how it would be possible to form fuzzy clusterings between object behaviours and local groups.

The advanced objectives were to enhance the clustering method by considering object and cluster scope. This was then to be evaluated accordingly to see whether these considerations could bring an improvement to the field. In order to carry out evaluation, it was decided that a selection of the best existing algorithms would be implemented. It was also decided that we would evaluate the algorithms using different dynamic object

behaviours, to represent various playing styles in games. We achieved the advanced objectives by extending the method used in TKM, which is to assign each object a circular threshold. We provided each object with a set of primitive shapes acting as their scope. The object then contains two variables; its *shape threshold* and *current location*. The object only issues an update to the server when its current location is not within its shape threshold. This greatly reduced the network overhead of OMBG to a fraction of what is used in TKM, giving excellent results.

II. Related Work

In this section, it was extremely important that we collected a sample of the best existing clustering techniques. Fortunately, due to the huge variety of clustering applications, there exist detailed overviews of recent advances in the clustering field (Kotsiantis & Pintelas, 2004). There also exist papers dedicated to finding the best current techniques categorised under the general approach used to solve the problem (Jain, Murty, & P.J.Flynn, 1999). We do not wish to repeat this work, but it is important to understand that we wish to abstract and modularise parts of the best algorithms so that we can hope to combine and extend them in the solution. This requires revisiting the best current algorithms as referenced in the clustering overview; however it also requires examining other algorithms which are more relevant to our distributed scenario.

An overview of clustering (Kotsiantis & Pintelas, 2004) outlines that algorithms can generally be categorised into four types of method. In hope of finding a good selection of abstracted clustering techniques for creating our hybrid algorithm, it is important that we visit each of these categories and critically analyse every part of the best algorithms which solve the clustering problem. We start by examining partitional methods, the first and most simple approach. We decided to use a partitional method as the foundation of our algorithm due to their excellent converging performance and also their space complexity being just $O(k + n$ where k is the number of clusters and n is the number of objects.

A. Partitional Methods

MacQueen presented k-means clustering in 1967. This simple algorithm forms the basis for many partitional approaches and it is still used in a large number of applications today due to its fast converging properties. Attractive properties of k-means, and other partitional algorithms, are that they do not require any fixed structures in memory; such as a dendrogram as used in hierarchical methods, and that they offer good benchmarking properties.

The k-means algorithm can be computed by the following steps (MacQueen, 1967):

1. Predetermine the number of clusters by ideas from the field of rate distortion theory (Sugar & James, 2003).
2. Randomly generate k cluster centroid positions: $m_1^{(1)}, \dots, m_k^{(1)}$
3. Assignment every object to the nearest cluster centres:

$$S_i^{(t)} = \left\{ x_j : \left| x_j - m_i^{(t)} \right| \leq \left| x_j - m_{i*}^{(t)} \right| \forall\, i^* = 1, \dots, k \right\}$$

4. Update centroid locations to the mean distance of all objects within cluster:

$$m_i^{(t+1)} = \frac{1}{\left| S_i^{(t)} \right|} \sum_{x_j \in S_i^{(t)}} x_j$$

5. Repeat steps three and four.

The worst case time complexity of the k-means method has recently been improved from $\Omega(n)$ to superpolynomial $2^{\Omega(\sqrt{n})}$ runtime (Arthur & Vassilvitskii, 2006). Both the runtime is efficient in clustering large data sets and also the space complexity is good: $O(k + n)$. This is because, like most other partitional methods, the algorithm only needs to obtain a single dividing in the data; rather than create a clustering structure. It is the message complexity, in terms of network overhead, which now becomes a key focus for our k-means analysis. The simple solution for implementing k-means, in a distributed environment, is to re-evaluate k-means for every object update. The problem with this approach is that the message complexity and processing of the algorithm soon becomes too high for large datasets, as each object tries to repeatedly send update messages to a single server. We also have to look at the quality of the clusters created from the k-means method. It is known through observation that k-means is sensitive to the initial seeding of the centroid positions, as the algorithm converges on a local optimum (Results: Figure 8). Other observations following a survey conducted about recent clustering methods also note the following about clusters produced with the k-means method (Kotsiantis & Pintelas, 2004). 1. They are spherical-shaped as clusters are only calculated based on Euclidean distance. 2. They are affected by noise. 3. Each object only belongs to a single parent cluster. 4. The size of k needs to be known before the algorithm runs. 5. All data must be known in advance producing results in a batch-like effect.

However, the assignment and update steps (three and four), provide a simple foundation to efficiently solve the clustering problem. In our solution we use this foundation and then build on top of it, by using and progressing components from other methods.

We now look at two of the most recent implementations of k-means: 1. The k^*-means algorithm (Cheung, 2003), which aims to solve the problem of k-means requiring a fixed input k, and only producing spherical clusters, and. 2. The Threshold-based k-Means Monitoring algorithm, TKM (Zhenjie, Yin, Tung, & Papadias, 2008), which addresses the problem of k-means producing a high network overhead in online environments. Our intention is to be able to combine the attractive properties of these algorithms without creating any undesired side-effects. In order to achieve this strategy, we need to re-examine the algorithms and extract the desired components which are useful to our scenario. We start with a closer look at the k^*-means algorithm.

The k^*-means algorithm produces outstanding performance while maintaining good cluster quality. This is achieved by considering ellipse-shaped data clusters as well as spherical-shaped ones. The algorithm also achieves good results without giving the correct pre-determined input k.

The k^*-means algorithm can be computed by the following steps (Cheung, 2003). *Note:* We have adjusted the algorithm on lines four and eight so that it progresses even when there are no changes in the centroids, and hence not terminating:

1. Assuming the number of clusters is $k \geq k^*$, randomly initialize k centroid positions: $m_1, m_2, \dots, m_k$
2. Randomly select an object x_t from the input set, and for $j = 1,2, \dots k$, let:

$$u_j = \begin{cases} 1 \text{ if } j = w = \arg\min_r \lambda_r \|x_t - m_r\|; \\ 0 \text{ otherwise} \end{cases}$$

where $\lambda_j = \dfrac{n_j}{\sum_{r=1}^{k} n_r}$ and n_r is the cumulative number of occurrences of $u_r = 1$.

3. Update the winning centroid m_w only by: $m_w^{(\text{new})} = m_w^{(\text{old})} + \eta(x_t - m_w^{(\text{old})})$
4. Repeat steps two and three. If it is the first run, start step five concurrently.
5. Initialise $\alpha_j = \frac{1}{k \text{ for } j} = 1,2,\ldots,k$ and let $\sum j$ be the covariance matrix (the change between values) of those data points with $u_j = 1$.
6. Given an object x_t, assign into a cluster j with a *winner-take-all* principle if:

$$I(j|x_t) = \begin{cases} 1 \text{ if } j = w = \arg\min_r \rho_r; \\ 0 \text{ otherwise} \end{cases}$$

with $p_r = [(x_t - m_r)^{\mathrm{T}} \sum_r^{-1} (x_t - m_r) - \ln(|\sum_r^{-1}|) - 2\ln(\alpha_r)]$

7. Update the winning seed point m_w only by:

$$m_w^{(\text{new})} = m_w^{(\text{old})} - \eta \frac{\partial R}{\partial m_w}\Big|_{m_w^{(\text{old})}}$$

8. Repeat steps six and seven.

Upon examination of the k^*-means algorithm, we actually see that it contains two separate processes which branch out at step four. The first process, steps one to four, is similar to standard k-means in which some large numbers of clusters distribute themselves across the objects. The second process, steps five to eight, assesses the clusters and decides a winning cluster. A *winner-take-all* principle then penalises the loosing clusters by pushing them away, resultantly giving ellipse-shaped formations.

The original algorithm only progresses to step five when the clusters stop moving, and then the algorithm terminates when this condition reoccurs at step eight. This behaviour is unacceptable for dynamic clustering as we want to continuously cluster the objects without terminating any of the processes. Also, it is unlikely that the centroids will ever stop moving in dynamic scenarios. Our modification of the algorithm means that the two processes run concurrently. The problem with this solution is that steps one to four will be attempting to evenly distribute the clusters by ball-shaped gravity, meanwhile steps five to eight will be attempting to unevenly stretch the clusters by ellipse-shaped gravity. This antagonistic behaviour is undesired as it means that the clusters will be caught between the two gravitational fields; rather than resting in ellipse formation.

Threshold-based k-Means Monitoring (TKM) is our second recent partitional method which looks at how a server may monitor objects using a k-means method (Zhenjie, Yin, Tung, & Papadias, 2008). The simple solution, which produces the best quality, is to re-evaluate k-means for every object update; however this means the network communication becomes too large. TKM attempts to reduce this by initially assigning every object x_i a threshold θ_i, such that x_i only needs to send a message to the server if its previous known location differs by at least θ_i. Upon receiving a message from an object, the server computes a new incremental k-means using an algorithm based on a deviation of hill-climbing, and then calculates a new set of threshold values. These new threshold values only need to be sent to a small section of the overall objects because most objects' thresholds remain the same.

TKM uses a separate threshold for each object because objects closer to their parent centroid will be able to move more freely without requiring cluster reassignment. Also objects travelling faster will require larger thresholds to prevent updates from occurring more frequently. TKM defines thresholds with a radius, giving objects the freedom to move arbitrarily within a circle. In our solution, we look to improve the method adopted by TKM by also considering the fact that object movement is usually not arbitrary; we consider a categorisation of object movement and social behaviour which relates to a primitive shape, acting as the new threshold.

B. Hierarchical Methods

In this section, we examine a small extract of the best current hierarchical methods, in hope of finding components which may be used alongside our k-means foundation, to increase the quality of our clusters. Hierarchical algorithms fall under two categories; agglomerative methods (which create clusters using bottom-up design), and divisive methods (which create clusters using a divide-and-conquer approach). A recent survey conducted on clustering techniques outlines the following properties of hierarchical methods (Kotsiantis & Pintelas, 2004). 1. They do not require the number of clusters to be known in advance. 2. They compute a hierarchy of clusters stored in a dendrogram (poor space complexity compared to partitional methods). 3. They allow for a good representation of results. 4. Results can be sliced to give a single flat partition.

We briefly examine some of the most recent and developed hierarchical algorithms which are currently available (Kotsiantis & Pintelas, 2004). We start with a closer look at the BUBBLE and BUBBLE-FM, due to the following properties (Ganti, Ramakrishnan, Gehrke, Powell, & French, 1999). 1. They consider the problem of clustering large datasets in a metric space rather than just a coordinate space (our intermediate objective). 2. They produce high quality clusters from a single scan over data. 3. They can handle noise effectively. 4. Abstract data can be collected from the required CF^*-Tree structure. 5. They do not require the number of clusters to be known in advance.

BUBBLE and BUBBLE-FM are instances of the BIRCH framework (Zhang, Ramakrishnan, & Livny, 1996). One of the key reasons for the success of BIRCH is from its underlying CF-Tree (Cluster Feature). A CF-Tree aims to solve the problem where objects being read sequentially into an environment typically need to examine every existing cluster, to decide which is the closest. The CF-Tree is an in-memory index, height balanced tree which reduces the computation for deciding the closest cluster to logarithmic time complexity, rather than linear.

Initially a CF-Tree has a fixed number of nodes, however due to its evolutionary properties, the tree will change in size as new objects are inserted. If we wished to consider a CF-Tree with our algorithm, we would need to extend the structure to handle dynamic data by creating a new algorithm for when objects are updated or removed from the tree. Removing or updating objects from a CF-Tree would need to make use of a recursive function which has the potential to merge two child clusters, from a leaf node, into a stronger single cluster. The advantage of using a CF-Tree with our algorithm would be the reduced computation on distance checks. However a CF-Tree has increased space complexity and only considers spherical-shaped clusters based on its threshold radius criterion.

BUBBLE and BUBBLE-FM extend the method adopted by BIRCH by considering general distance space. Computing in distance space can be very expensive (for example multidimensional scaling computes in $O(n^2)$ complexity), whereas it is possible in $O(n)$ complexity with coordinate space. BUBBLE-FM uses an algorithm called FastMap (Faloutsos & Lin, 1995) which reduces general distance calculations to complexity $O(k)$ on large datasets, regardless of input size n. With our algorithm, we consider using FastMap as a replacement for our distance calculations, if they need to be extended beyond coordinate space. It is worth noting that FastMap is not solving the problem of reducing general metric distance calculations, it is only presenting a practical method of reducing it such that the quality of results becomes the limiting factor.

One of the most recent hierarchical algorithms for clustering large data sets is GRIN, which offers an incremental and scalable algorithm in $O(n)$ complexity (Chen, Hwang, & Oyang,

2002). An important feature which sets GRIN apart from other algorithms is that it is not sensitive to the initial distribution of objects in the environment. Also, its incremental approach means that it can easily handle huge and continually growing datasets. GRIN is based on the ideas behind gravity theory, which gives it the following important properties (Oyang, Chen, & Yang, 2001). 1. It is not biased towards spherical clusters 2. It does not suffer from the chaining effect (where clusters pull together and dominate multiple natural groupings). 3. It can have a lower time complexity or a lower space complexity in comparison to other hierarchical algorithms.

Clustering based on the ideas behind gravity theory works by analysing the velocities of clusters. When two clusters collide (which means the distance between the two objects is less than the lumped sum of their radii), they merge to form a higher level cluster (Oyang, Chen, & Yang, 2001). There is also a resistance within the environment which guarantees the eventual merging of all clusters into a single parent cluster.

C. Density-based and Grid-based Methods

The survey conducted on recent advances in clustering (Kotsiantis & Pintelas, 2004) outlines two final categories of clustering algorithm: density-based and grid-based methods. We do not consider using density-based methods for our *foundation algorithm* due to the following two reasons. 1. They typically rely on a nearest neighbour function which does not scale well in large datasets. 2. Cluster quality is restricted by the performance of the distance function, as used in retrieving neighbours.

Despite these disadvantages, density-based methods usually offer superior quality with the following advantages. 1. They do not require the number of clusters to be known in advance. 2. They are not biased to particular cluster shapes or formations. 3. They are not affected by noise.

On observation, we can see that these attributes are near the exact opposite of partitional k-means methods. Recent research has looked into combining k-means with a popular density-based algorithm, DBSCAN, to create a new algorithm BRIDGE with the following advantages (Dash, Liu, & Xu, 2001). 1. Increased speed and scalability. 2. Ease in setting a density threshold for density-based clustering. 3. Improved k-means quality without being effected by noise. However the algorithm still requires k to be known beforehand.

Density-based approaches are similar to that of the previously described gravitational methods, using another tree-like structure to build a hierarchy of clusters. With our solution we only consider using them as an option to boost cluster quality.

Grid-based Methods are an interesting approach used to reduce, but not solve, the clustering problem. They work by partitioning the environment into a determinable structure and then clustering the density of objects mapped to this structure (Kotsiantis & Pintelas, 2004). By doing this, the performance of grid-based methods remains constant however the quality of clusters becomes variable. In a large virtual online environment, many players may form subgroups in a single crowded location. Relating back to the determinable structure of grid-based methods, the algorithm will only perceive a single dense cell for such crowded locations. It will not be able to perform clustering within the cell, which will give an incorrect result at that level of accuracy. However, if the resolution of the grid is smaller than that of the player's in-game perception, then there will never be such a case where more than one cluster is required per cell, because the user will want to see all objects. In this case, a grid is suitable for our algorithm.

III. SOLUTION

With our work, we hope to address the problem where clustering algorithms have large message complexity. The latest methods used in TKM produce low message complexity, however they also produce poor quality clusters in superpolynomial time, as an undesired side-effect. Our solution, OMBG, is an extremely efficient algorithm in terms of time complexity, space complexity, and message complexity while producing excellent quality clusters. To construct our method, we extended on the ideas which are explored in the related work section, starting with k-means as our foundation algorithm. It is important to understand that the majority of our solution is incremental as we are attempting to bring new concepts into the clustering field. However we have provided ourselves with a large resource of components which can be progressed and used to solve any pitfalls encountered during the process of development.

A. *Foundation Work*

We started with an implementation of the k-means method which would later be used to evaluate and compare our algorithm with. The simulation tool was programmed in C++ using OpenGL to render large quantities of dynamic objects within the environment. We also designed our architecture such that we run features of our algorithm in parallel by using synchronisation and multithreading.

Using a modern PC, with an i7 CPU clocked at 4 × 2.66 GHz, our simulation tool can process and render 100,000 dynamic objects while implementing k-means at 60 *FPS* and 1,000,000 dynamic objects at about 10 *FPS*. We tested k-means with 10,000,000 objects however the rendering speed is generally 1-2 *FPS* and with k as small as 50, the update speed is only about once every ten seconds with such huge quantities of data to display.

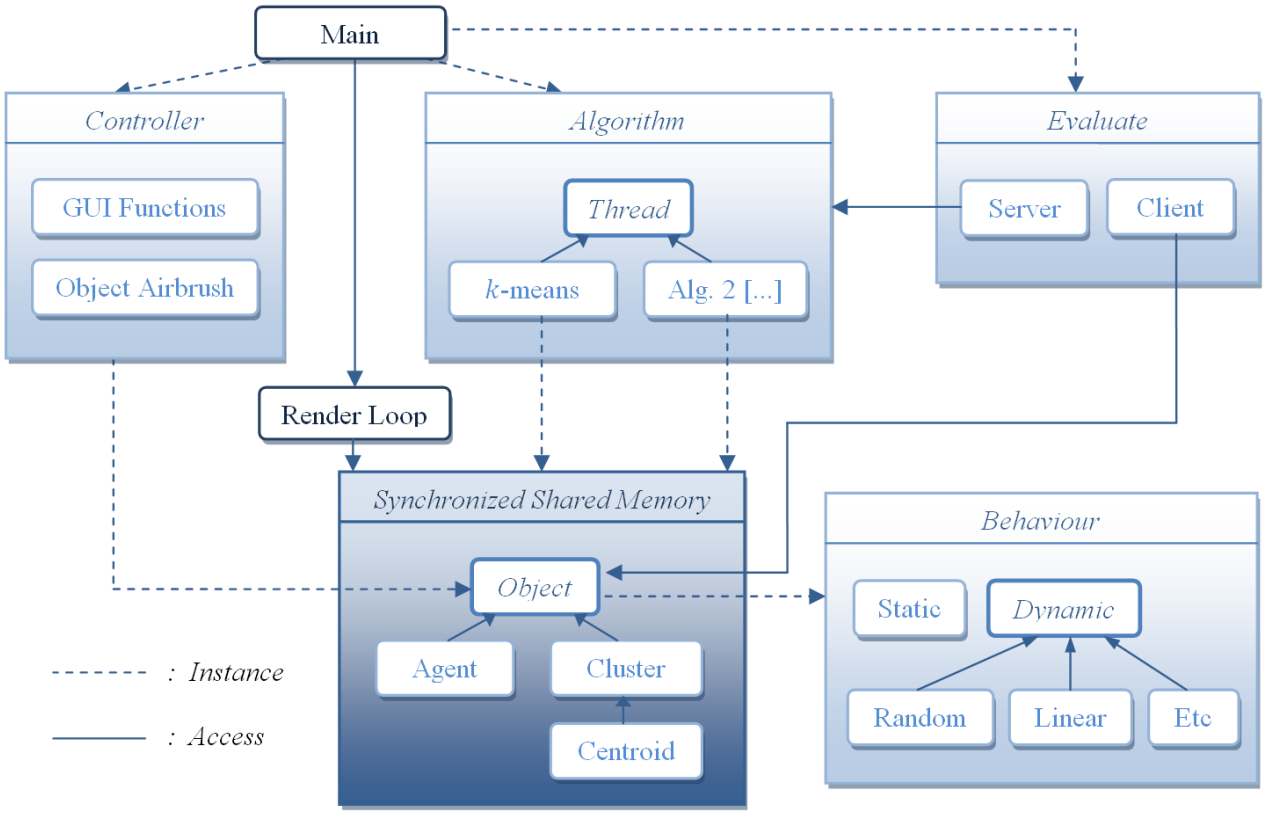


Figure 1. Abstract of our simulation environment architecture.

The final architecture of our tool has to achieve the following: 1. Virtualise a client-server model. 2. Perform high-performance and responsive rendering of reactive objects. 3. Implement potentially parallel algorithms while sharing memory with the render loop. 4. Dynamically allocate and de-allocate memory to each object. 5. Attach non-Euclidean behaviour to each object, and. 6. Perform evaluation. The interconnectivity between parts of our simulation environment, which achieves these criteria, can best be described using the component diagram in Figure 1.

Initial observations from our k-means foundation clarify several problems outlined in our related work: 1. Even with correct input k, clusters are incorrect with non-spherical natural groupings (Results: Figure 8). 2. Noise is still present and clustered. 3. Parallel k-means produces speckled cluster boundaries (Results: Figure 7b). 4. Performance is adequate, but can be improved, and 5. Network overhead is too large. However, we have achieved our basic objective by implementing a simple clustering method.

The main priority was to address points one to three by improving clustering quality. Our related work outlines the following algorithms with improved cluster quality: k^*-means, BUBBLE, GRIN, DBSCAN, and BRIDGE. From this selection, we refined our search by process of elimination from our intermediate objective, adding the handling of non-Euclidean metrics, and GRIN resulted in having the most advantages.

The problem with improving cluster quality is that every object has to check every cluster and decide which one to join based on some existing information stored on the cluster. Let us consider these three scenarios:

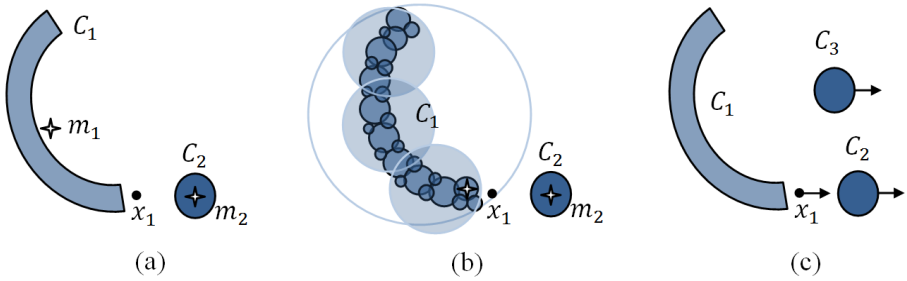


Figure 2. Three complicated clustering scenarios.

In Figure 2a, object x_1 has to decide between joining clusters C_1 and C_2. If we decide to join the clusters based on the closest centroids; m_1 and m_2, then x_1 will incorrectly choose m_2 and join C_2 even though C_1 is closer. The question is deciding what to store in the clusters which lets x_1 choose C_1 over C_2. GRIN samples a number of points within each cluster and forms a hierarchy of subclusters, as shown in Figure 2b. Object x_1 traverses the dendrogram of each cluster until it finds the nearest child cluster, and then it joins the child cluster's parent. The problem with this approach is that it still requires an expensive dendrogram structure, and also the speed of the algorithm varies depending on the sampling process. However, a major advantage of the GRIN algorithm is that object updates can be added to the structure incrementally, and this is something we hope to keep. Figure 2c shows another problem with clustering: If object x_1 inherits similar *behaviour* to C_2, it would be better for x_1 to join C_2 rather than C_1. We classify object b*ehaviour* as an identifier of a predefined concept, such as a movement trajectory or a characteristic. Example behaviours could represent reoccurring movement trajectories; such as running in a straight line, or staying around a fixed location; or they could represent non-Euclidean characteristics such as whether an object is *social* or *aggressive*. However, we cannot simply cluster based on behaviour, else

x_1 may join C_3 instead of C_2. Another reason for our classification of object behaviour is that we can use it to reduce the message complexity of our algorithm as outlined in heading *D*.

B. *Clustering Structure*

Clustering algorithms which require every object to check against every cluster must run in at least $O(n * k)$ complexity; and so we decided to use a clustering structure to reduce this and also increase the quality. The problem with using a dendrogram is that they have relatively high space complexity according to the minimum number of children per leaf, and they have access times of $\log n$. A grid structure would be a better solution, with constant access time and space complexity according to some resolution. As outlined in our related work, a grid would be feasible in clustering game environments providing that each cell is smaller than the viewing scope of each player. The viewing scope is either a frustum defining a *3d* field of view, or a circle defining a *2d* object's circle of perception. Given a circular scope of diameter D, we can calculate the maximum cell size which fits into D using the following formula: $Maximum\ Cell\ Size = \sqrt{(D^2/2)}$

This is calculated by applying Pythagorean Theorem to calculate one of the equal sized edges of the right-angled triangle whose hypotenuse is D. Alternatively with a viewing frustum, we can calculate the maximum cell size by subtracting the *far plane* depth from the *near plane* depth in relation to the camera. By deciding to use a grid, our design now branches as we classify player objects as *Agents* and introduce a new kind of parent object, *Cells*. This level of abstraction allows us to compute nearest neighbours in constant time complexity and also iterate the entire grid in constant time complexity, which will be extremely useful later when determining cluster formation. In our algorithm, we developed an extremely fast method of merging and splitting cells to form clusters based on their most popular agent behaviour densities. We named this method Object-Mapped Behaviour-Grid, or *OMBG*. Let us consider two natural groups of agents, and within these groups, two different agent behaviours represented by shades of blue:

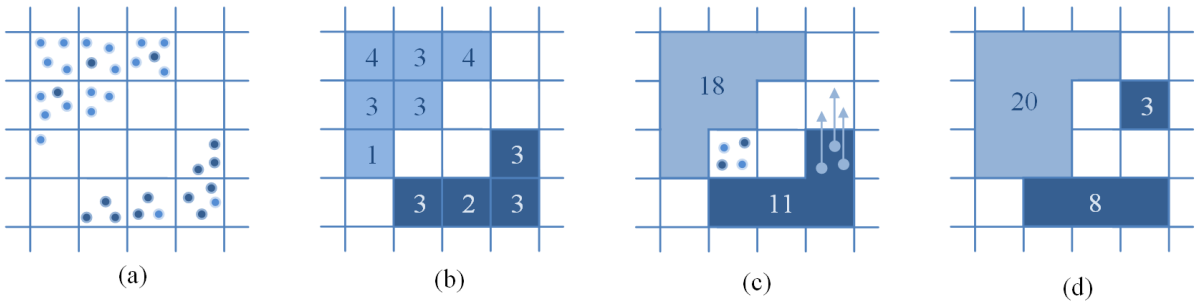


Figure 3. Cell mapping in OMBG.

This process starts by counting all the agents and their corresponding behaviours within each cell T_j. The most popular behaviour then emerges as the *winning-cell-behaviour,* B_{win}. After this, T_j uses a *relationship function* that determines which cluster it should join based on both B_{win}, and the proximity of nearby cells of the same type as B_{win}. Figure 3a and b, show how agents of the most popular behaviour type are counted and mapped to cells. Figure 3c then represents two scenarios: 1. Agents of equal behaviour distribution are caught between two clusters; their corresponding *Cell* gets mapped to the strongest cluster (18 > 11 = 18). 2. Agents move away from their cluster which makes their new cell no longer have enough agents to be part of the parent cluster; the vacant *Cell* leaves its parent cluster. The final diagram, Figure 3d shows how the split has updated their parent cluster values, and the Cell

occupied by the three agents is attempting to form a new cluster of its most popular behaviour type. At this stage, we can also see that there is contact between the two large clusters, however because they are of different behaviours we may not wish to merge their cells. We cover this in our relationship function.

C. *Relationship Function*

The relationship function is the core of OMBG. It is called once sequentially for each cell T_j, unless we run the algorithm in parallel. It determines whether cell T_j *joins* an existing cluster, *leaves* its current clusters, or *creates* a new cluster. The function requires a threshold θ which is used as a circular scope determining each cells neighbours, and it requires knowing the minimum number of agents required for remote cells to create their own cluster: A_{min}. The relationship function can be computed for our algorithm as follows:

1. For *Cell* T_j inspect all neighbour *Cells* N_j within threshold θ.
2. For each neighbour $N_j \in \theta$, find the most popular behaviour the parent cluster of Cell N_j'.
3. Label this behaviour B and label the number of agents of type B in the cluster of N_j' as $|B_A|$.
4. Let T_j *leave* its current cluster.
5. If $|B_A \in T_j| < |B_A \in N_j'|$, then
   If $|B_A \in N_j'| > A_{min}$, then T_j *joins* the Cluster of N_j'
   End If
   Else If $|B_A \in T_j| > A_{min}$, then T_j *creates* a new cluster.

It is important to understand that we wish to constantly iterate the relationship function for every cell in the grid. We label the currently selected cell from this iteration as T_j. Step 1 starts by retrieving a list of $T_j's$ neighbour cells. For example if $\theta = 1$, this would return the cell above, below, left, and right of T_j. Step 2 then examines each neighbour's parent cluster, and identifies the most popular behaviour type: i.e. the one with the most objects. Step 3 labels this behaviour B and from this cluster, we label the total number of agents in $B's$ cluster as $|B_A|$. Steps 4 and 5 are where the action occurs; we start by forcing T_j to leave its current cluster. Step 5 then compares the *winning population* of T_j with the *winning population* of $T_j's$ best neighbour. By *winning population,* we mean the total number of agents of behaviour B within the cell's cluster. Lines 1 and 2 in Step 5 mean that a cell only joins a new cluster *if* it has a neighbour with a winning population that is greater than its own winning population and greater than the minimum number of agents required to form a cluster: A_{min}. The final line in step 5 means that a cell creates a new cluster if it has no neighbour with a greater winning population, and if its own winning population is greater than the minimum number of agents required to form a cluster.

D. *Agent to Cell Mapping*

Previously we mentioned that the first stage of our algorithm was to count all the agents and their corresponding behaviours within each cell T_j. By storing the current and previous Cell Id for each agent, we can reliably map agents to cells in constant time *without counting,* by using the following *client-sided* algorithm for each agent:

1. Wait until the *Agent* state changes, (it gets added to the environment, removed or if it the agent's location updates).
2. Self evaluate current behaviour as B. (Calhoun, Stahovich, Kurtoglu, & Kara, 2002)
3. Get the current Cell Id, T_{cur} from the agent's location. *Note: This is in constant time.*
4. If $T_{cur} \neq T_{prv}$
   {
   $|B_A \in T_{prv}| = |B_A \in T_{prv}| - 1$
   $|B_A \in T_{cur}| = |B_A \in T_{cur}| + 1$
   $T_{prv} = T_{cur}$
   }
5. Remove agents colour or mark agent as *noise*.
6. If $|B_A \in T_{prv}| > A_{min}$
   Colour agent as the same colour as its current cells parent cluster.

Each cell stores a list of values, which represent the total number of agents inside the cell of each behaviour type. Step four checks whether the agent's current cell is equal to the agent's previous cell; if they are different, it decrements its corresponding behaviour total for the previous cell and increments its corresponding behaviour total for the current cell. The last statement in step four sets the previous cell to the current cell so that in the next iteration of the agent update, the agent can reliably release behaviour totals even if it has moved to a completely different cell. We finally colour the agent in Step 6 if the winning population of its current cell is greater than the minimum number of agents required to form a cluster.

E. *Overview and Improvements*

With this method applied on the client, we remove the need to have a mapping algorithm on the server, and thus our entire algorithm only consists of a single iteration of the relationship function for each cell. Because the number of agents does not influence the cell count, our entire algorithm is now hence reduced to constant time complexity.

Our message complexity is currently almost identical to that of TKM; a message is only sent to the server when an agent changes cell in step four. In TKM, a message is only sent to a server when a client leaves a circular threshold. However, we can now extend our method and achieve a smaller message complexity than TKM by changing the initial *if statement* in step four of the client agent-cell mapping algorithm. Our replacement for the 'if statement' is:

If $B_{cur} \neq B_{prv}$ {
  $\forall$ *Cells* $T_j \in B_{prv}$ decrement $|B_A \in T_j|$
  $\forall$ *Cells* $T_j \in B_{cur}$ increment $|B_A \in T_j|$
  $B_{prv} = B_{cur}$
}

We recall that agents have the ability to self evaluate their behaviour. We now check whether the agent's current behaviour has changed from its previous behaviour; if it has changed, we send a *single* message to the server updating the totals for the current cells which collide *under* that behaviour *shape*. It is important to understand that lines two and three can be achieved in a *single* message containing a list of cells for the server to update. By this new concept, we are reducing the message overhead from that of repeatedly sending a

single cell *or threshold* to that of a group of cells which construct a primitive shape at the cost of a little extra self evaluation processing on each client.

The self evaluation processing requires keeping a buffer of agent locations and periodically mapping this to an object called a *Shape*. Each shape has an ID called *behaviour* (*B* in our algorithm), and contains a list of cells. In the following figure, we represent some example relative lists of cells for their corresponding behaviour name.

Figure 4. Self-evaluated behaviours.

We can use this to construct a message to the server as follows:

$$\mathrm{M}\langle\, 'L', prv\ \{36,37,38,39\},\ cur\ \{20,21,23,27\}\,\rangle$$

Where 'L' is the agent's evaluated behaviour, prv contains a list of previously affected cells (the numbers represent each cell's *id*), and cur contains a list of the currently affected cells (which should look similar to other shapes of that behaviour in the cluster).

Recognising geometric primitives is beyond the scope of this report; however there exist many efficient methods which deal with this field of computing, such as recognising multi-stroke symbols (Calhoun, Stahovich, Kurtoglu, & Kara, 2002). There is also recent research which deals with categorising and transforming shapes under a standard, which proves useful for implementing similar techniques (Hammond & Davis, 2004). However, we leave the process of categorising and defining object behaviours as primitive shapes an open research topic. In our implementation, we demonstrate the reduction with five behaviours; *Gentle* Central, *Swarming* Central, *Linear* Running, *Noise* and *Random* Running.

F. *Implementation and Testing Issues*

Due to the incremental and modular nature of our design, we did not encounter any major implementation or testing issues with the solution. Prior to using synchronisation, we experienced multiple crashes as objects were created and destroyed. However, following synchronisation, we have an extremely stable tool which has been tested running multiple large allocations of millions of objects and reallocating them accordingly for 24 hours without any runtime errors.

IV. Results

We compare OMBG with k-means, for its traditional benchmarking properties and also with TKM, as this is the most recent algorithm that yields the best solution for reducing network overhead. To provide comprehensive coverage and fair testing scenarios, we use different testing strategies for both *time complexity* and *message complexity*.

In each strategy, we also test different object distributions and multiple dynamic object behaviours. The reason for this is mainly to see how our method reduces message complexity in comparison to TKM. When analysing cluster quality, we also look at how giving the correct input k and the incorrect input k compares with our method (which does not require k).

A. *Time Complexity*

Our first and most important testing strategy is to see how quickly our algorithm converges and completes an iterative cycle comparatively. In *k*-means and TKM, an iteration is when both the assignment and update steps have completed, and in OMBG this is when the relationship function has been called once for every cell in the grid. To test this fairly, we created three scenarios; *a*, *b*, and *c*. Scenario *a* features randomly moving objects which we increased from 100 to 4,000,000 while collecting the number of iterations each algorithm produced every 10 seconds. In scenarios *b* and *c* we manually created between 100 and 1,000 natural groupings of objects for every 1,000 objects that we added. Scenario *b* contained natural groupings of static objects, whereas scenario *c* consisted of natural groupings of dynamic objects from one of five behaviours. Our results are plotted as follows:

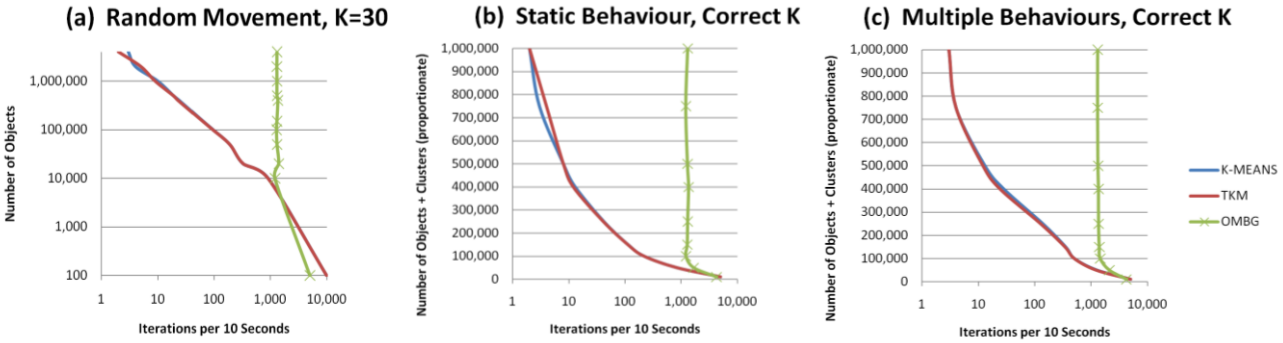


Figure 5. Runtime of *k*-means, TKM and OMBG.

Scenario *a* shows that as the number of objects increases, the runtime of *k*-means and TKM decreases accordingly. In a gaming environment, this means that the server would eventually lag behind with huge numbers of objects giving incorrect clusters. However, OMBG remains constant irrelevant of the number of objects in the environment. We predict that any slowdown in OMBG is caused by the processing priority and availability of the thread in which it runs on. Scenarios *b* and *c* show similar results, however with *k* increasing, *k*-means and TKM suffer additional penalties while OMBG stays in constant time complexity.

B. *Message Complexity*

The second most important testing strategy is to see how many messages are sent between objects and the server in different clustering scenarios. Our simulation tool allows us to measure time complexity and message complexity at the same time by virtualising the client-server model. Because of this, we were able to share scenarios *a*, *b*, and *c* as when measuring time complexity. Our results are plotted as follows, and described on the next page:

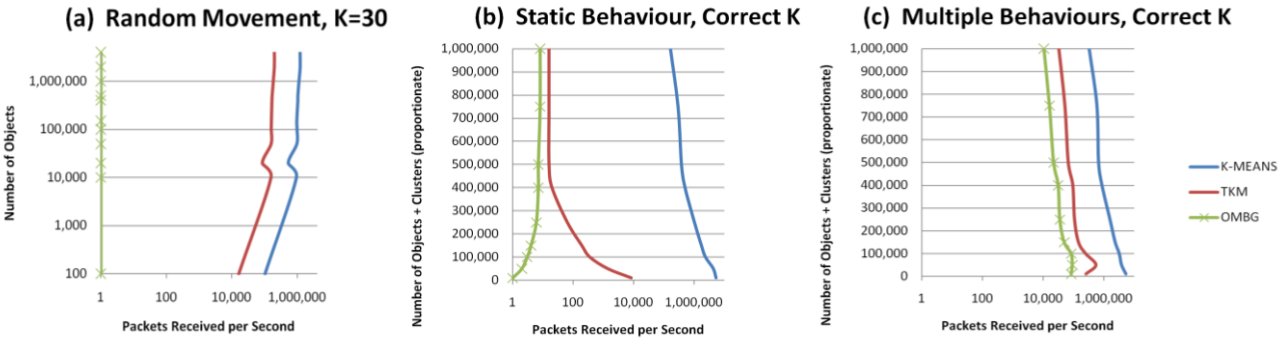


Figure 6. Communication overhead of *k*-means, TKM and OMBG.

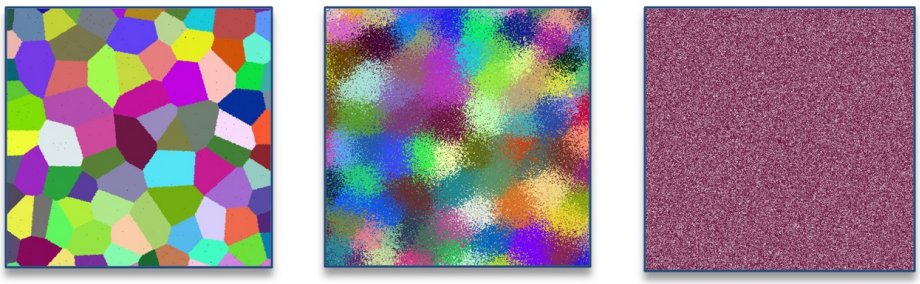
In *k*-means, a message containing each object's location information is repeatedly sent to the server. In TKM, a message is only sent to the server when new objects are added, removed or when an object leaves a threshold. And in OMBG, a message is only sent to the server when an object leaves a shape which represents its behaviour.

Scenario *a* shows that *k*-means sends millions of packets to the server every second, while TKM seems to only send a fraction of this. This is because objects are generally moving at the same speed and leaving their threshold proportionally. OMBG does not send a message to the server until object behaviour changes. In this case, we only have a single behaviour which produces no natural groupings; resultantly OMBG does not send any messages to the server. Scenario *b* shows *k*-means and TKM both receiving a reduced number of packets as the number of objects increases. This is due to the algorithm runtime slowing as the number of objects increases which limits the number of packets that can be received per second. OMBG is not affected by the algorithm runtime and thus the number of packets received increases only as new objects are added to the environment. Unlike scenario *a*, OMBG can cluster static behaviours and thus a single message is sent to the server for each packet. This is the reason for why we see some traffic in graph *b*. Scenario *c* shows the most crowded and busy situation. Objects are moving between clusters, changing behaviours, and forming new clusters in increasing quantities. We can see that *k*-means has identical worst-case message complexity as in the two previous scenarios. TKM has a similar result to scenario *a,* as most objects will eventually leave their threshold depending on how much they are moving. This is the worst-case for OMBG as we are constantly changing object behaviours. However, OMBG still outperforms TKM because primitive shapes more accurately predict movement trajectories than a circular threshold.

C. *Cluster Quality*

We look at the quality of clusters produced by each algorithm in a selection of different object behaviours and distributions. We only compare OMBG with *k*-means as the quality of clusters produced by TKM is near identical to *k*-means.

We first observe 10,000,000 evenly distributed moving objects, similar to scenario *a* in the time and message complexity analysis. In this scenario, there are no natural groupings of data and hence the optimum solution would be to have no clusters. However, the following three screenshots from our simulation tool show that *k*-means always produces clusters:



(a) Sequential *k*-means with incorrect *k*.

(b) Parallel *k*-means with incorrect *k*.

(c) *k*-means with correct *k* value = 0.

Figure 7. Evenly distributed objects in sequential and parallel algorithms with incorrect *k* and correct *k*.

Output *a* shows *voronoi cells* with tight cluster boundaries. This is because *k*-means does not consider eliminating noise; and with an incorrect value for *k*, the algorithm forces *k* groupings from the evenly distributed data. Output *b* shows a similar effect, however the *k*-means algorithm is now running in parallel with the moving objects. Because of this, objects

move away from their cluster before *k*-means completes an iteration, resulting in speckled cluster boundaries. Output *c* shows the result when running *k*-means with the correct input value for *k*. All of the objects are still visible because *k*-means cannot classify them as noise. The correct result is to classify the evenly distributed data as noise and not cluster or display it. OMBG achieves this, as shown in Figure 8; no agents are capable of filling cells with groupings greater than minimum number of agents required the form a cluster, and thus all cells are classified as noise and nothing is displayed. This is the optimum result.

We must now test our algorithm using different behaviours in *natural groupings* of data. We cover five of the most difficult clustering scenarios, and compare OMBG with *k*-means in each case. Figure 8 shows a table with a description of each scenario on the left, and it shows two screenshots from our simulation tool running both algorithms, on the right. The clusters are represented by colours whereas the natural groupings are represented by dense areas of objects. Noise is very difficult to see; hence in the first row we have manually highlighted every object black glow to make it more apparent:

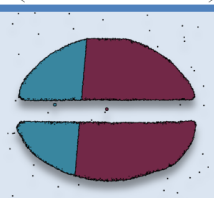
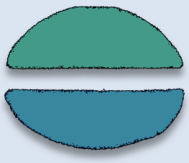
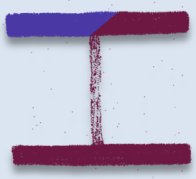
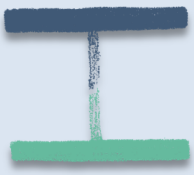
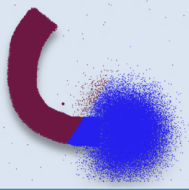
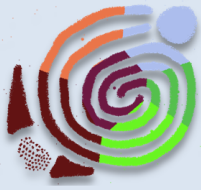
Scenario description and results for each row.	*k*-means: correct *k* (screenshot of tool)	*OMBG* (screenshot of tool)
In this first scenario, we show two clusters of data amongst some noise. We have highlighted the noise to make it visible. *k*-means cannot remove the noise and also performs incorrect clustering due to it only being able to consider spherical distances.		
This scenario shows how OMBG reacts to the *chaining effect* where one cluster dominates in mildly linked scenarios. We can control the migration threshold (*scope)* in our algorithm to produce an excellent result as shown.		
Here we can see that OMBG can elegantly handle a case of clustering which BIRCH and GRACE fail in (Chen, Hwang, & Oyang, 2002). In this simple scenario, *k*-means incorrectly clusters every grouping.		
This scenario shows how OMBG handles the problem presented in Figure 2*b* & *c*; Given two clusters of different ***dynamic behaviours***, our algorithm can distinguish between them even where contact is strongly present.		
This final scenario shows that OMBG is able to cluster extremely difficult scenarios with objects all of the ***same behaviour***. The two entwined spirals are correctly clustered and the speckled grouping, *bottom-left*, is recognised as a different cluster due to our scope threshold.		

Figure 8. Comparing OMBG with *k*-means and giving *k*-means the correct input *k*.

The screenshots from our algorithm show that it produces the optimum result in each of the five scenarios, whereas k-means does not produce any such results even with the correct value for k. This excellent outcome shows that OMBG can elegantly handle even the most complicated scenarios of static and dynamic object distributions.

V. Evaluation

We now reflect back to the original research question: 'Can we improve distributed clustering methods, and implement a new algorithm, by modularising and extending existing techniques, and considering object and cluster scope?' We answer this by examining the strengths and limitations of OMBG.

A. Strengths

OMBG is the first algorithm of its kind which performs well, produces excellent quality clusters, and has relatively low message complexity. Such results were produced by adopting methods from recent algorithms and creating a new hybrid class of solution. This uses techniques from grid-based methods, density clustering and TKM. We also introduced the idea of categorising behaviours as primitive shapes which resultantly showed an improvement in the reduction of network overhead, which we believe an important step in the progression of the clustering field. The notion of behaviour classification also works on a conceptual level; we can use it to consider restricted paths, or maps within games. For example, we could have behaviours such as "In Castle Area", or "Within Room" which only group with other objects of the same type. There are many other possibilities which game designers can consider while using non-Euclidean metric behaviours. I.e. social interaction between objects can now be clustered, allowing distributed servers to partition high-traffic objects in lesser volume than low-traffic objects. As a result of the simplicity and modularity of our design, such flexibility with behaviours can be directly managed *centrally* within the relationship function of our server algorithm.

B. Limitations

A limitation of OMBG is that it depends on a new and open research area; the categorisation of object behaviour as primitive shapes. We validate the need for this new topic by showing success in our results in defining six behaviours; static, gentle movement, fast movement, linear running, teleporting noise, and random-moving noise. Having decided upon which behaviours to classify, the clients require a shape to match a buffer of movement locations or trajectories with. This matching process is an existing field of computing, and we reference two papers which can be used to effectively achieve this (Calhoun, Stahovich, Kurtoglu, & Kara, 2002), and (Hammond & Davis, 2004). A second limitation of OMBG is that the algorithm requires two inputs; a grid *resolution size* and a *scope threshold* size. All our results were calculated using a scope $\theta = 1$ and a resolution of 100×100 cells. However, we found it easy to set the variables based on the hardware of the machine. We proposed a method for finding the *maximum cell size* in a gaming environment. Once this is calculated, it is possible to set θ to 1, and then divide the *world size* of the game environment by the *maximum cell size* to get an optimum resolution value; hence eliminating this limitation.

C. Approach

Our incremental and modular approach gave us the opportunity to use and extend existing components from other algorithms. However, during the design, we found that every option

in using existing methods resulted in undesired side-effects as outlined in clustering evaluations (Kotsiantis & Pintelas, 2004) and (Jain, Murty, & P.J.Flynn, 1999). We therefore decided to construct a completely new hybrid of density, grid, and TKM. We extended these methods by adding *scope* and *primitive shape* concepts, which we would have been unable to do without allowing for incremental changes. The key to success in our approach was in providing ourselves with a solid foundation *sandbox* to work in and test ideas; this was our simulation tool. The extensive feature list of the simulation tool consisted of; multithreading, a simulated client-server model, GPU-acceleration, synchronisation, and modularity of design. These features gave us the flexibility to change our algorithms; insert new objects (such as Cells), and observe different methods in any given scenario.

Modularity in our design is clear with our *algorithm*; we have two components; the relationship function; which runs on the server, and the object-cell mapping; which runs on each client. These two components, although dependent on each other, are physically separate with the only required communication being in packets of data. We also do not explicitly state that the relationship function has to be embedded within a loop. The reason for this is also due to the modular approach adopted; in modern computers running multiple processors, we can allocate each processor to running the relationship function on a section of cells in the grid. This allows us to reduce the *constant* time complexity to a fraction of the total number of processors, when running in parallel.

If given the chance to repeat the work, we would not change this approach. In hindsight, having seen that improvements in the clustering field come from new areas, it may have been better to spend more time in examining references from areas other than computer science, such as biology and physics, in hope of finding more creative solutions to the clustering problem.

VI. Conclusion

The importance of this work can be stressed by the three main contributions which we have brought to the field of clustering:

1. We introduced a new hybrid class of algorithm which has constant time complexity, produces excellent quality clusters, and generates the lowest comparable message overhead.
2. We show that by mapping objects to behaviours, we can reduce message overhead to a lower value than that of TKM, the most recent and successful algorithm in this field.
3. We validate the need for a new research topic which identifies a suitable method for classifying object behaviours as primitive shapes.

To summarise our solution, we examined the most successful recent clustering methods (Kotsiantis & Pintelas, 2004) and attempted to modularise and extend their key components. We constructed an adaptable simulation tool to act as a solid foundation for testing future algorithms, starting with a simple implementation of k-means. By following a process of elimination, we then filtered only those components useful for our scenario.

Following this, we used an incremental and modular design process to merge concepts from grid clustering, density clustering, and TKM, with our own idea of behaviour categorisation. This produced a low network object-mapped behaviour-grid algorithm with exceptional performance and quality, which we named OMBG. Although this was initially designed for distributed game environments, OMBG can also be used in other applications even without a client-server model. Due to its extremely low network overhead and constant time complexity, these could include mobile computing, search engine indexing, biological data analysis, and physics simulations. The object-behaviour mapping also makes the

algorithm suitable for complex applications such as analysing multi-agent systems and human movements.

A suitable direction for future work would be to create a selection of different primitive shapes which represent object behaviours. These can then be evaluated in hope of gaining a deeper understanding of why some behaviours work better than others in certain scenarios. A practical direction for future work would be to merge the recently developed standard for shape recognition (Hammond & Davis, 2004) with the object-cell mapping part of our client algorithm. This would allow application developers to instantly have access to a large collection of shape recognisers allowing for more flexibility in shape construction and evaluation.

REFERENCES

Arthur, D., & Vassilvitskii, S. (2006). "How Slow is the k-Means Method?" *Proceedings of the twenty-second annual symposium on Computational geometry* (pp. 144-153). Sedona, Arizona, USA: ACM.

Calhoun, C., Stahovich, T. F., Kurtoglu, T., & Kara, L. B. (2002). "Recognizing Multi-Stroke Symbols." *AAAI Spring Symposium on Sketch Understanding* (pp. 15-23). Pennsylvania: AAAI Press.

Chen, C.-Y., Hwang, S.-C., & Oyang, Y.-J. (2002). "An Incremental Hierarchical Data Clustering Algorithm Based on Gravity Theory." *Advances in Knowledge Discovery and Data Mining: 6th Pacific-Asia Conference. 2336*, pp. 237-250. Taipei, Taiwan: Springer Berlin / Heidelberg.

Cheung, Y.-M. (2003). "k*-Means: A New Generalized k-Means Clustering Algorithm." *Pattern Recognition Letters , 24* (15), 2883-2893.

Dash, M., Liu, H., & Xu, X. (2001). "`1+1>2': Merging Distance and Density Based Clustering." *Proceedings of the 7th International Conference on Database Systems for Advanced Applications* (pp. 32-39). Hong Kong, China: IEEE Computer Society.

Faloutsos, C., & Lin, K.-I. (1995). "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets." *Proceedings of the 1995 ACM SIGMOD International Conference on Management of data* (pp. 163 - 174). California: ACM.

Ganti, V., Ramakrishnan, R., Gehrke, J., Powell, A., & French, J. (1999). "Clustering Large Datasets in Arbitrary Metric Spaces." *Proceedings of the 15th International Conference on Data Engineering* (p. 502). Los Alamitos, CA: IEEE Computer Society.

Hammond, T., & Davis, R. (2004). "Automatically Transforming Symbolic Shape Descriptions for Use in Sketch Recognition." *Proceedings of the Nineteenth National Conference on Artificial Intelligence* (pp. 450-456). Cambridge, MA: AAAI Press.

Jain, A., Murty, M., & P.J.Flynn. (1999). "Data Clustering: A Review." *ACM Computing Surveys , 31* (2), 264-323.

Kotsiantis, S., & Pintelas, P. E. (2004). "*Recent Advances in Clustering: A Brief Survey."* University of Patras, Department of Mathematics. WSEAS Transactions on Information Science and Applications.

MacQueen, J. (1967). "Some Methods for Classification and Analysis of Multivariate Observations." *Proceedings of the 5-th Berkeley Symposium on Mathematical Statistics and Probability* (pp. 281-297). Berkeley: University of California Press.

Ng, B., Lau, R. W., Si, A., & Li, F. W. (2005). "Multiserver Support for Large-Scale Distributed Virtual Environments." *IEEE TRANSACTIONS ON MULTIMEDIA , 7* (6), 1054-1065.

Oyang, Y.-J., Chen, C.-Y., & Yang, T.-W. (2001). "A Study on the Hierarchical Data Clustering Algorithm Based on Gravity Theory." *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery* (pp. 350 - 361). London: Springer-Verlag.

Sugar, C. A., & James, G. M. (2003). "Finding the Number of Clusters in a Data Set: An Information Theoretic Approach." *American Statistical Association , 98* (463), 750-763.

Zhang, T., Ramakrishnan, R., & Livny, M. (1996). "BIRCH: An Efficient Data Clustering Method for Very Large Databases." *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data* (pp. 103-114). Montreal, Canada: ACM Press.

Zhenjie, Z., Yin, Y., Tung, A. K., & Papadias, D. (2008, June 20). "Continuous k-Means Monitoring over Moving Objects." *To appear in TKDE , 20* (9), pp. 1205-1216.

Facial Recognition with Eigenface Decomposition and Neural Networks

Student Name:

Supervisor Nam

Submitted as part of the degree of Computer Science to the

Board of Examiners in the Department of Computer Science, Durham University.

Abstract –

Context/Background: Facial Recognition in Computer Vision is concerned with the detection and classification of human faces in images. A well-known approach which gives satisfactory results involves encoding images using eigenface decomposition as an input to single neural network classification, and extending to using neural network ensembles can further improve results.

Aims: To investigate if a solution using eigenface decomposition and a single neural network can solve the problems of face detection, face identification, and gender classification. To optimise the solution for each problem by calibrating the neural networks and extending the solution using a neural network ensemble. To evaluate the robustness of the solution to image blur and noise.

Method: Images from a standard database of face and non-face images are encoded as a linear combination of the best *k* principal components (eigenfaces) of a subset of the face images. To solve each problem, a single neural network is trained to appropriately classify encoded images, and the accuracy optimisation of the network is investigated, before using it in an ensemble to determine if any further accuracy can be gained. The robustness of the solution to image blur and noise is investigated by observing change in accuracy rates when these factors are increased in test images.

Results: Using a non-trivial set of face images (i.e. with variation in face characteristics and orientation), maximum accuracy rates are 98.08% for face detection, 94.81% for face identification, however the solution could not solve the gender classification problem on this set of images. The only accuracy gained for use of an ensemble was for face detection, with a gain of 0.25% over a single neural network. The solution was very robust to image blur, and somewhat robust to image noise.

Conclusions: The solution is very appropriate for the problems of face detection and face identification, but inappropriate for gender classification. Neural network ensembles remain of interest, but a simple ensemble of clones operating with 'bagging' training method and majority vote system did not produce satisfactory accuracy gains. The robustness of the solution to poor-quality images is indicative of its practical worth.

Keywords – Facial Recognition, detection, identification, gender classification, eigenface, neural network, ensemble, blur, noise

I. Introduction

Facial Recognition (FR) is an increasingly active research area in Computer Vision, and can be broadly defined as the automated detection and classification of human faces in images. The core problems are face detection and face identification, and there exist various other interesting problems such as gender, age, and expression classification.

The inherent difficulty of FR problems stems from the complexity and subtle variability of human faces coupled with the fact that most interesting applications involve unpredictable

variations in factors such as lighting, image noise and blur, and face orientation. The general requirements of any solution are therefore to solve problems as accurately as possible under 'normal' conditions, with a minimal degradation in accuracy as these factors vary from the norm. Unsurprisingly, the development of solutions to better satisfy these requirements remains the primary focus of research in the area.

A. Applications of Facial Recognition

The earliest applications of FR were primarily in automated surveillance systems restricted to use by large organisations such as Governments due to the large cost of the required computing and imaging hardware. However in very recent years a combination of factors including the fall in cost and rise in performance of hardware, the growth of the internet, and an intensified research effort have led to a rapid expansion in the range and user-base of FR applications.

At present, automated security and surveillance remains one of the key application domains, with FR technology increasingly being incorporated into 'biometric' identification systems at airports and banks worldwide. Another important domain which illustrates the growing accessibility of the technology is human-computer interaction, with widely commercially available applications including face-based login systems on personal computers and face-tracking webcams. In addition, the internet progressively opens up new frontiers for FR applications. Currently, the search engine *Google* offers an image search which restricts results to face images, and it is possible that web image searches for specific individuals will be developed in the future.

B. Nature of Facial Recognition Problems

FR problems are essentially pattern classification problems, where an image (pattern) is analysed and classified based on its contents. For example in face detection the image is classed as 'face' if it contains a face and 'non-face' otherwise. In order to perform such classification, images must be encoded numerically such that there is a basis for mathematical comparison which can be used to group images into classes.

A numerically encoded image can be represented using a finite n values. For example, an intensity image i of resolution (pxq) is encoded in its 'raw' form using pq pixel intensity values. If these n values are arranged in some uniform manner into an n-dimensional (n-D) column vector, this gives the coordinates of the unique 'location' of the image in an n-D space which represents the space of all possible images using the same encoding. Figure 1 illustrates an analogy which is useful in conceptualising such an n-D space.

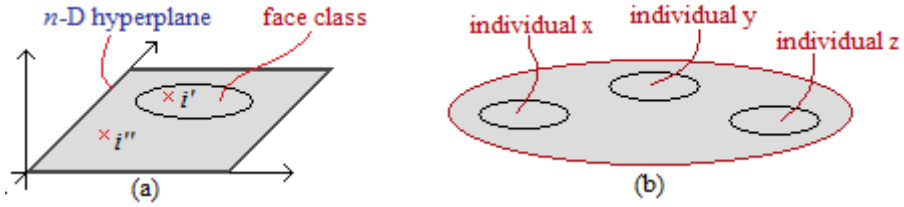


Figure 1. Visualisation of classes and sub-classes of images in n-D space
(a) The face class is the 'area' of n-D space containing face images: Image i' is a face, image i'' is not
(b) The face class can be divided into sub-classes, for example classes of images of a particular individual's face

Just as a 2D vector gives the coordinates of a point on a 2D plane, an image represented as an n-D vector can be described as a 'point' on an n-D 'hyperplane'. Related images have related encodings and therefore have related coordinates in n-D space thus forming clusters of points

on the hyperplane. A class of image can therefore be defined as a function which describes the bounds of such a cluster, such that an image is a member of the class if and only if it is located within these bounds. Hence the difficulty in solving any FR problem lies in the accurate estimation of the bounding functions of the required class(es).
For clarity the class bounds in Figure 1 are depicted as neat 'circular' functions. However in reality face classes are likely to have highly complex bounding functions due to the myriad of possible variations in face images (e.g. face size, pose, lighting to name but a few). Accurate estimation of these bounds is thus non-trivial and only increases in difficulty as n increases. Importantly, n is typically *very* large for 'raw' images since each pixel represents a separate dimension.

It is possible however to reduce the difficulty of the classification problem. By controlling variables such as pose in input images, some of the complexity of face classes can be controlled, and most crucially the dimensionality of the problem can be reduced by projecting images from the original n-D image space to a lower dimensional subspace which is specific to FR. The derivation of such a subspace and the method of projection can be described as an image *encoding* method. By preserving as much relevant information as possible from the original image, an encoding can be used as the input to image classification, the success of which depends on how efficiently and effectively the encoding represents the original image for the purposes of FR.

As discussed in Section II, this general two-part approach of encoding images prior to classification is the most popular and successful in FR research to date.

C. Project Objectives

The project will develop a general Facial Recognition solution using eigenface decomposition (for image encoding) and neural network(s) (for image classification). The solution will be applied to the problems of face detection, face identification, and gender classification. For each problem, there will be an investigation into optimising the accuracy of the solution, initially using a single neural network for classification, then extending to using a neural network ensemble to investigate any further accuracy gain that this may yield. Finally, the robustness of the method will be investigated by observing any change in accuracy rates of the optimised solution for each problem when using test images with added noise and blur, respectively.

Formulated as a series of questions, the project can be summarised as follows: Can an FR solution which encodes images using eigenface decomposition, as an input to single neural network classification, be applied to solve the problems of face detection, face identification, and gender classification? If so, how can the solution be optimised for each problem to increase the accuracy rate? Can this accuracy rate be further improved by extending from single neural networks to a neural network ensemble for classification? Finally, how robust is the solution to increased blur and noise in test images?

II. Related Work

Research in FR has been active for approximately 40 years, over which time the rate of work produced has accelerated. Although it is impossible to detail more than a small subset of this work, this section will summarise, analyse and compare the major techniques which have demonstrated the most success in practice.

A. Input Images

As a Computer Vision problem, the type and quantity of input images used for FR are critical considerations. Single-intensity images (single greyscale photographs) have traditionally been used, but hardware advances have given rise to alternative imaging techniques in more recent research. In early research [1][2], it is arguable that the use of intensity images was forced by lack of viable alternatives, yet their use has remained popular even in very recent work for sound reasons. Aside from the informal observation that humans (can) use only light intensity as an input to extremely successful FR, the low cost of hardware, capture, and processing in addition to near-instantaneous and non-invasive capture give the single-intensity image a practicality unrivalled by alternative imaging modes.

The argument for alternative imaging techniques is that greater volumes and/or different types of image data can provide more information useful for FR than is possible with a single-intensity image. Such techniques generally fall into two categories: Those using multiple intensity images, for example from multiple cameras [3] and video sequences [4], and those using altogether different imaging hardware, for example infra-red cameras [5] and 3D laser scanners [6].

Much of the work using single-intensity images can also be applied to multiple and infra-red intensity images and vice-versa. Conversely, work using 3D imaging introduces a somewhat different 3D approach to FR. Such work has shown promising results in the lab; however the laser scanner is the only currently available hardware which can produce a sufficiently accurate 3D image of a face, therefore such techniques are impractical or infeasible for use in most real-world applications.

Although alternative inputs to single-intensity images certainly warrant discussion, their associated approaches to FR solutions are beyond the scope of this paper. From here on, the term *image* will refer to an intensity image and the remainder of this section will discuss only work using this input mode.

B. Face Encoding

By far the most popular general approach to FR solutions is to encode input images to reduce their dimensionality prior to classification. A raw image with n pixels can be described using exactly n intensity values and is thus n-D. For an image of useful pixel resolution, n is large, resulting in a high initial dimensionality. The goal of an image encoding method is to reduce this dimensionality by extracting only the information relevant to FR and discarding the rest.

i. Feature-Based Approach

One intuitive approach to encoding a human face from an image is to construct a mathematical model of the face based on its physical features (eyes, nose etc).

This approach appeared in the earliest FR research, where faces were encoded using the ratios of geometrical relationships (such as Euclidean distance) between features located in the image. Such work was conducted as early as 1966 by Bledsoe [1], using manual feature location and ratios of Euclidian distances, and 1973 by Kanade [2], progressing to automated feature location and a more comprehensive set of distance, area and angle ratios.

A more recent and sophisticated feature-based encoding method is ‘Elastic Bunch-Graph Matching’ (EBGM), introduced in 1999 in [7]. In this method a face is encoded as a graph or *grid*, where nodes represent features and edges represent distances between them. Progressing from using just distances, each feature node is labelled with a ‘Gabor jet’. A jet taken at a

pixel describes the mathematical behaviour of surrounding pixel intensities, thus can describe a facial feature in an image. A grid describes the 'local' detail of one face, and grids describing related faces can be stacked into a 'bunch graph' to provide 'global' information about a class. This is shown in Figure 2.

The use of geometrical ratios in describing faces has some advantages, such as tolerance for variations in face size and position within the image; however there is little tolerance for changes in expression or face rotation as these factors can distort the ratios. In addition, a set of ratios is useful only for basic identification, but cannot be used to classify on factors such as gender, race, or age, limiting the use of the early methods to fairly trivial applications.

The added sophistication of EBGM comes from the use of jets, which provide much richer local information and also facilitate global information through the use of bunch graphs. This means that EBGM is far less sensitive to changes in expression and rotation, and can be used for far more general classifications than early feature-based methods.

However, the necessity for either cumbersome manual or inaccurate automated feature location introduces issues which place all feature-based methods at a critical practical disadvantage to alternatives (to be discussed shortly), and such concerns are directly raised in [7]. In addition, research in human FR such as [8],[9], has shown that individual features and their geometrical relationships do not provide adequate information to explain the high FR performance of humans, suggesting this approach is inherently naïve.

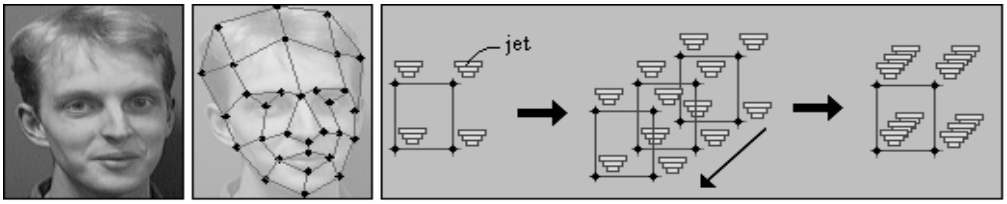


Figure 2. EGBM (a) original face image (b) features located and grid constructed (c) Jets on nodes can be combined into bunch graphs to describe a class

ii. Statistical Approach

Conclusions about the inadequacy of the feature-based approach to extract the information truly relevant to FR give rise to a somewhat different approach to face image encoding. By treating input images as linear, this approach uses statistical analysis to extract a set of local and global mathematical 'features' from face images. Almost counter-intuitively, the hypothesis is that such abstract mathematical features are more relevant to FR than human-recognisable face features.

The 'eigenface' method was the first example of a statistical approach to face image encoding, and formed the basis of the first truly practical FR systems. The underlying ideas of the method were introduced by Sirovich and Kirby in 1986 [10], and were later developed and applied to FR in a famed 1991 paper by Turk and Pentland [11]. The method (along with variants) remains in use even in current research; with [11] being the most widely and frequently cited paper (according to SCOPUS, WoS) across the FR literature.

The full detail and mathematics of the method will be elaborated upon in section III, as there is not sufficient space here. As a very brief outline, the eigenface method is based on Principal Component Analysis (PCA), a statistical tool used to extract the 'directions' of greatest variance in a data set. PCA is thus applied to a set of face images to extract these vectors of greatest variance, called eigenfaces, which describe the principal components (most significant features) of the set of face images, such that in general any image of the

same dimensions as the original face images can be 'built up' from these eigenfaces. The accuracy with which an image can be 'built up' from (or decomposed into) different combinations of the eigenfaces determines the properties of the face in an FR context. Thus, an image can be represented as a vector of coefficients describing its eigenface decomposition which is of much lower dimensionality than the number of pixels in the original image. This concept is illustrated in Figure 3, taken from [12].

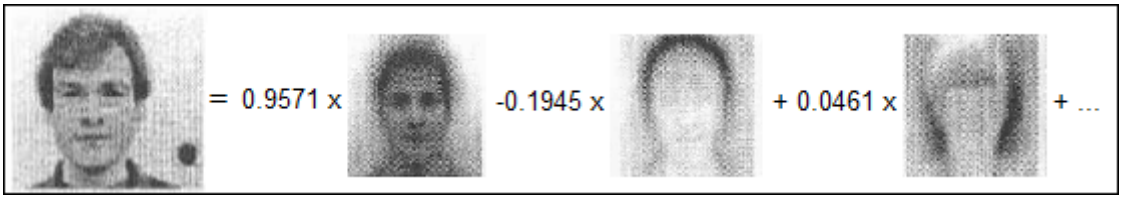


Figure 3. Decomposition of a face image into eigenface components (taken from [12])

The advantages of the eigenface method are significant, because the encoding is essentially a mapping from a high-dimensionality original image space to a low-dimensionality eigenface space which preserves a great deal of relevant FR information. This means that dimensionality of the classification problem is greatly reduced without limiting the range and accuracy of possible classifications, the encoding can be automated quickly and with absolute correctness, avoiding practicality issues seen in the feature-based approach, and all images (of the correct dimensions) can be encoded, facilitating face detection. Furthermore, research in human FR such as [8],[13], has drawn specific parallels between human FR mechanisms and the eigenface method, suggesting that an eigenface decomposition encoding contains information more relevant to FR than encodings produced by alternative methods.

Despite its benefits, however, the method also has shortcomings. Eigenfaces are derived from the variance of images in the set **F** of face images from which they are derived, therefore they can be said to 'blindly' represent only the global and local features of faces in **F**. This means that the encoding will poorly represent any deviation from the 'norms' of **F**, so the method is fairly sensitive to variations in many unpredictable factors such as face size, position, rotation, expression, lighting, and so on. Representing all possible variations of face images in **F** is unfeasible and thus this constitutes a significant practical issue with the eigenface encoding method.

An alternative statistical encoding method which tries to reduce the problem of sensitivity to lighting variation seen in the eigenface method is the fisherface method, first explicitly introduced in 1997 [14]. The method is very similar in nature to the eigenface method, except that Linear Discriminant Analysis (LDA), rather than PCA, is used to derive a lower-dimensionality image space. Because PCA is optimal on intensity variance, this can produce unwanted eigenfaces which merely represent changes in lighting rather than changes in features. LDA on the other hand maximises not on inter-image variance but on inter-class variance by making use of *a priori* knowledge about the classes images belong to. By projecting training images from c classes from image space to a c-1 dimensional hyperplane such that classes are separated into groups on the hyperplane, a much lower dimensionality subspace is defined. A new image can then be projected into the subspace in the same way as the training images, where it should be 'located' within its class bounds.

In principle, the fisherface method is less sensitive than the eigenface method to variations in lighting and similar factors which distort intensity variance within a class of face image such as expression, because inter-image variance is 'ignored'. However there are various ways in which the eigenface method can be made less sensitive to such variations. Most obviously, the face images used to derive the eigenfaces can be controlled for variations

in lighting and expression, and otherwise the removal of the first few eigenfaces can remove much of the unwanted variation due to lighting [14].

A major disadvantage of the fisherface method is the complete dependence of the method on adequate generality of the training data. As mentioned, the eigenface method suffers when the set of training faces is insufficiently general since variance is reduced, but the problem is much more serious for the fisherface method. The problem is most likely when attempting to linearly separate two very 'similar' classes which are 'close together' in image space, such as male and female faces. If an unrepresentative sample is used for either of the classes in the training images, then the classes will not be correctly separated, thus accurate classification is impossible and the method breaks down. Though in general, accurate estimation of class bounds using the eigenface method is more difficult than using the fisherface method, the modelling of classes is also less susceptible to dramatic error since images in eigenface-space mirror the 'natural' classes of original image-space and so it is arguable that the eigenface method is preferable to the fisherface method provided classification is performed accurately.

C. Face Classification

The second 'stage' of a general FR solution is the classification of images based on their encodings. If *n*-D images have been encoded as *k*-D vectors, then this involves accurate estimation of class bounds in *k*-D space. Clearly, the class(es) requiring estimation depend on the FR problem of interest, and the accuracy with which they are estimated defines the accuracy of the solution.

The most straightforward classification approach uses a measure of distance (usually Euclidian) between image encodings viewed as *k*-D vectors to measure for 'similarity' either directly between images [2], or between an image and a class, using a threshold distance from the mean image of a class to determine class membership [11].

The distance-based classification method is undeniably simple, and essentially approximates class bounds as 'circular', using the threshold as a 'radius'. Nonetheless, the method showed fairly accurate classification for face detection and identification, under controlled conditions, using eigenface encoding in Turk and Pentland's landmark 1999 system [11]. However, such a simple approximation of class bounding functions using thresholds places an upper limit on both the accuracy and range of possible classifications.

The difficulty of accurately and precisely estimating the bounding functions of image classes using 'algorithmic' methods such as the distance-based methods stems from the difficulty of generalising these functions when very little can be assumed about them. It is suggested that this problem might be solved with machine learning techniques, and the neural network (NN) is one such technique which has been applied in work such as [15].

Again, a detailed description of NNs will be given in section III, but briefly, an NN is a model of computation which can 'learn' to approximate a function given training data and ideal output, and can therefore learn to approximate the bounds of an image class in k-D space given a set of training images and their classes. The advantages of using NNs over algorithmic approaches are numerous. An NN can approximate a continuous function with high accuracy and precision based entirely on the data, without having to simplify class bounding functions with assumptions, and can be easily re-trained for any classification without any change in implementation, giving an accuracy and flexibility unrivalled by algorithmic approaches. Furthermore, multiple NNs can be combined into NN *ensembles*, which can give even further gains in classification accuracy compared to singular NNs [16].

The NN approach is not without shortcomings, however. Though theoretically very powerful, the actual classification power of NNs is limited by fixed computational resources;

a problem which increases with the dimensionality of the classification problem. In addition, NN training is a subtle process for which there is no defined process for accuracy optimisation. Therefore, for more 'straightforward' FR classifications such as face detection, it may be argued that algorithmic approaches are still preferred due to their simplicity.

III. Solution

The solution developed to solve the three problems of face detection, face identification and gender classification is presented in this section. As is typically the case in FR, a general solution was developed and then applied to and optimised for each specific problem. To begin, an overview of the design of the solution is given, and then each stage is described in detail. Then, the optimisation of the general solution to each problem and the practical issues of the solution implementation and testing are discussed.

The overview of the design of the solution is illustrated in Figure 4. Following from the most successful related work, the solution was based on the general 'two-stage' approach of encoding images prior to classification, but the solution is decomposed further here into four stages giving a specific description of how it was implemented. Giving a very brief outline, these four stages can be grouped into two distinct sections of the solution: *training* and *testing*.

The training section is performed once, and involves organising a database of images to use as input, encoding the images using eigenface decomposition, and using a specific set of encoded training images, training the 'NN-unit' to perform the appropriate classification to solve a specific problem. As outlined in section I the solution will start by using a single NN and will extend to using an NN ensemble for classification, so the term 'NN-unit' is used to describe both where a general explanation is required.

The testing section can be performed arbitrarily many times once the training section is complete, and involves retrieving a pre-encoded test image from the database (an image which has not been used in the training section), giving it as input to the NN-unit, and analysing the output from the NN-unit which determines how the image has been classified.

When the solution is applied to a specific problem, the only factor which *must* be altered is the training of the NN-unit to perform classification in such a way to solve that problem. For example, when applied to face detection the NN-unit is trained to classify encoded images as either 'face' or 'non-face'. Additionally, both the eigenface decomposition encoding and the NN-unit training can be optimised for a specific problem. However the four general stages defined in figure 4 are followed no matter which problem the solution is applied to, therefore a description of each stage can adequately explain how the solution is applied to any FR problem.

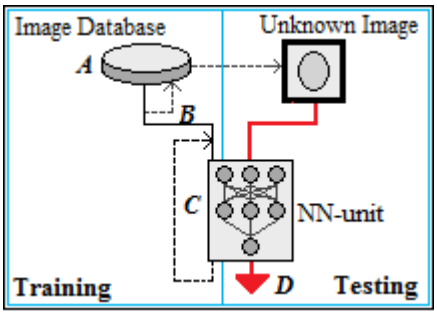


Figure 4. Overview of Solution Design. ***A.*** Compilation and organisation of image database ***B.*** Image encoding using eigenface decomposition ***C.*** NN-unit training ***D***. NN-unit testing

A. ***Compilation and Organisation of Image Database***

As implied in the previous section, the chosen input to the solution was 2D single-intensity images. In practice the source of these images may vary between applications, but the focus here is on the solution itself and not the input source. Therefore, input images were acquired from appropriate sources and stored in a local MySQL database for convenience. This database is referred to as the *image database*. In addition to a set of face images, a set of non-face images were also required since face detection was one of the problems of interest.

The choice of face images to use as input to the solution was of high importance. Numerous 'standard' sets of face images which are specifically compiled for FR research are available on the internet, and offer significant advantages over a 'random' collection for the testing and evaluation of an FR solution. Standard sets generally have a structured scheme of variation in factors such as face size, position, rotation, expression, background, and lighting, which creates a framework for structured and controlled testing.

The face set chosen for use was the '*AT&T Database of Faces*' (from AT&T Labs, Cambridge), a sample of which is displayed in figure 6. The set contains 40 individuals with 10 face images per individual, giving 400 face images. In general, the faces are centralised and fill the image. Across individuals there is variation of ethnicity, gender, hair style, facial hair, and the wearing of glasses, and within individuals' images there are significant variations of expression, and minor variations of face position and orientation.

The AT&T Database of Faces was an appropriate choice for a number of reasons. Crucially, the number of individuals and images per individual, and the good inter- and intra-individual variation, make the set very useful for testing the solution. This is because it gives confidence that any test results will scale to the 'general' case and are not local to only certain 'types' of faces. This set, unlike some other prominent standard sets, also contains an adequate number of female face images (40 across 4 individuals) to be feasibly used to test gender classification. In addition, the set has been used widely across FR work, which as explained will be useful in evaluating the solution against alternatives.

Though perhaps not as critical, the choice of non-face images was also important. The actual objects in non-face images can be considered arbitrary, but the images must be similar on factors such as the size of object, background, and brightness to the face images used, to ensure that face detection classification is done on the face or non-face in an image and not on more distinct unrelated disparities. For this reason, the chosen non-face image set was a subset of the '*Caltech256 Dataset*', which is a standard set of object images for use in Computer Vision. As shown in Figure 6, these non-face images are ideal since the object(s) of interest are central, fill the image, and are set against a plain background.

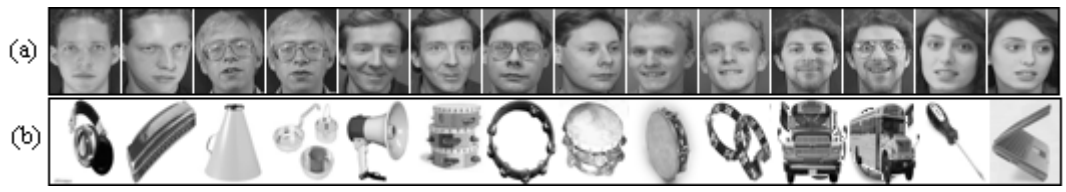


Figure 6. Samples from the image database.
(a) Faces from *AT&T Database of Faces* (b) Non-faces from *Caltech256 dataset*

Having acquired sets of face and non-face images, they were stored in the image database, which could then be organised according to the problem of interest. Basically, for a given problem the images were logically divided into a training set and a test set, the former being

used for eigenface computation and NN-unit training, the latter being used for NN-unit testing.

B. Image Encoding Using Eigenface Decomposition

Following the compilation of the image database, the next stage was to encode all images in the training and test sets, since each would be required eventually. The image encoding method used in the solution was *eigenface decomposition*, which involved two steps. Firstly, the computation of eigenfaces from the faces in the training set, then the decomposition of every image into a linear combination of these eigenfaces (the encoding).

Let F be the set of face images in the training set. Since a standard set of face images was used, no pre-processing was required; however in general the images in F *must* be of identical dimensions, and the faces within them should be central and of the same size. The faces in F were chosen to be a representative sample of the faces in the image database as a whole, to increase the likelihood that the eigenfaces computed from F would capture the variance across all the faces in the image database.

The computation of eigenfaces of F is done using PCA, which involves a series of straightforward linear algebra operations. Let M be the number of face images in the training set. Each image is initially represented as a matrix of pixel intensity values of dimensions (m x n). Each matrix is converted to a column vector of dimensions (mn x 1) by 'stacking' the n columns of the matrix. This gives a set of M face image vectors, Γ. Next, the mean face image, φ, is computed using Eq. (1). The image vectors in Γ are then normalised by using Eq. (2) to give a set of normalised image vectors, Φ, where Φ_i is the distance of the face Γ_i from the mean face φ. Next, the covariance matrix **C** is computed using Eq. (3). The eigenvectors of **C** are the eigenfaces of F. However, the computation of the eigenvectors of **C** is unfeasibly expensive for useful image sizes, since **C** is of dimensions (mn x mn). This problem is resolved by first noting that $\mathbf{C} = \mathbf{A}\mathbf{A}^T$, where the matrix **A** of dimensions (mn x M) is the result of compounding (concatenating) the normalised image vectors in Φ. Then, as shown in Figure 7, by using the general eigenvector equation given in Eq. (4) to compute the eigenvectors $\mathbf{v}_i$ of the much smaller matrix $\mathbf{A}^T\mathbf{A}$ of dimensions (M x M), the M 'best' eigenvectors $\mathbf{u}_i$ of $\mathbf{A}\mathbf{A}^T$ and therefore of **C** can be computed as $\mathbf{u}_i = \mathbf{A}\mathbf{v}_i$. The mn eigenvectors of **C** represent mn orthogonal directions of variance in the face images of F, and the eigenvalue associated with each eigenvector gives the magnitude of variance. The M computed eigenvectors have the highest eigenvalues of all the eigenvectors in **C**, meaning they represent the M directions of greatest variance in F, and are thus the M 'best' eigenfaces. Finally, each of the M eigenfaces are normalised such that $\| \mathbf{u}_i \| = 1$.

$$\varphi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i \tag{1}$$

$$\Phi_i = \Gamma_i - \varphi \tag{2}$$

$$C = \frac{1}{M}\sum_{j=1}^{M}\Phi_j\Phi_j^T \tag{3}$$

$$B\mathbf{x}_i = \lambda_i\mathbf{x}_i \tag{4}$$

$$
\begin{aligned}
&\text{From Eq. (4):} && (\mathbf{A}^T\mathbf{A})\mathbf{v}_i = \lambda_i\mathbf{v}_i \\
&\therefore && \mathbf{A}\mathbf{A}^T\mathbf{A}\mathbf{v}_i = \mathbf{A}\lambda_i\mathbf{v}_i = \lambda_i\mathbf{A}\mathbf{v}_i \\
&\therefore && \mathbf{C}\mathbf{A}\mathbf{v}_i = \lambda_i\mathbf{A}\mathbf{v}_i \\
&\therefore \text{ From Eq (4):} && \mathbf{C}\mathbf{u}_i = \lambda_i\mathbf{u}_i \quad \textit{where} \quad \mathbf{u}_i = \mathbf{A}\mathbf{v}_i
\end{aligned}
$$

Figure 7. Showing that $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ have the same eigenvalues, and eigenvectors related by $\mathbf{u}_i = \mathbf{A}\mathbf{v}_i$ [12]

Once the M best eigenfaces have been computed, the K eigenfaces corresponding to the K largest eigenvalues (again, the K most significant eigenfaces) are kept and the remainder are discarded. K is a variable which depends on the problem being solved. The eigenfaces with the highest eigenvalues represent the most variance in F and are therefore the most global face features. The less significant eigenfaces represent directions of less variance and therefore more local features such as the eye and mouth shape of a particular individual. Thus K is determined by the level of 'detail' needed for optimal classification in a particular problem.

When K eigenfaces have been computed, the encoding of each image from the image database by eigenface decomposition is very straightforward. First, an image must be of identical dimensions to the images in F. The image is then represented as a column vector Π_i in precisely the same manner as the images in F, and normalised by subtracting the mean image of F, φ. The normalised image Ψ_i can then be represented as a linear combination of eigenfaces as seen in Eq. (5).

$$\Psi_i = \sum_{j=1}^{K}(w_j\mathbf{u}_j) \quad \textit{where} \quad w_j = \mathbf{u}_j^T\Psi_i \tag{5}$$

Clearly, this linear combination can be represented by K 'eigenface coefficients', with the coefficient for $\mathbf{u}_j$ given by the value w_j as defined in Eq. (5). By representing these K coefficients in a vector, we have a K-D encoding for the original image Π_i describing the decomposition of the image into K eigenface components with different weights (which may be positive or negative and determine the extent to which the eigenface 'contributes' to the image). Each image in the image database was encoded using exactly this method.

C. NN-Unit Training

Having encoded all images in the image database using eigenface decomposition, the remainder of the solution is concerned with the classification of the encoded images (and thus the images they represent) in order to solve the problem of interest.

The method of classification used in the solution was based on NNs. Firstly, the standard approach of using a single NN was adopted, then the solution was extended to using an NN ensemble. The NN ensemble was essentially a collection of replicas of a single NN which had been pre-optimised for the given problem. Therefore, the description of the singular NN architecture and training method will cover both 'types' of NN-unit, and a brief additional explanation of the specifics of the NN ensemble will follow.

An NN is a computational model based on the biological brain, which has a software implementation in the solution. The field of NNs is vast and far beyond the scope of this paper, therefore all that can be described here is the NN architecture used in the solution, illustrated in Figure 8. This architecture and configuration are typical however, thus the properties described apply to most NNs.

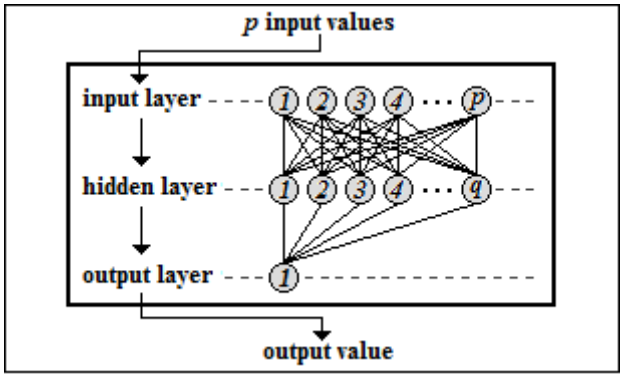


Figure 8. NN architecture used in the solution

As shown in Figure 8, the NN is a network of 'neurons' which are connected by weighted edges, or 'links'. A single neuron is capable of applying an 'activation function' to an input value to give an output value. The activation function used is a common sigmoid function known as the 'logistic function', given by Eq. (6).

$$f(x) = \frac{1}{1+e^{-x}} \tag{6}$$

Clearly, individual neurons perform a trivial operation, but the underlying idea of the NN is to combine many neurons to achieve a more complex 'global' behaviour. In this NN architecture, this global behaviour is defined by the links between neurons. Following the standard *feed-forward* architecture, the neurons are organised into three *layers*: input, hidden, and output. The links are only between consecutive layers, and are only 'forwards' in direction, as indicated by the arrows in Figure 8, such that each neuron in a layer is connected to each neuron in the next layer (but *not* the other way round) by a unique and direct link. Each link has a *weight*, which is a real positive or negative value.

The NN operates as follows: A k-D image encoding is given as input such that each of the k values goes to a specific input neuron (p = k). In each input layer neuron, the logistic function is applied to the input to give some output, which is then 'sent' to the hidden layer down *every* link leaving each neuron. Crucially, the value sent on each link is multiplied by the weight of the link. Each hidden layer neuron thus receives p values which are then summed to give the input to that neuron. The process repeats for the hidden layer, and eventually the single output layer neuron receives q values, which are summed and passed to the logistic function to give a single output value, which is the output of the NN.

From the above process it is clear that the NN can approximate a function that takes an image encoding as input and gives a value indicating the class as output. The function approximated depends entirely on the link weights, and the logistic function at each neuron is simply used to 'encourage' values near to 0 or 1, since each of face detection, face identification, and gender classification have been formulated as 'yes/no' classifications.

Since changing link weights changes the function approximated by the NN, it should be possible to change them such that they can approximate a class bounding function in k-D space. The critical advantage of NNs is that the link weights can be changed automatically in a 'training' process to better approximate any class bounding function, given some training data, i.e. some image encodings and their classes (which must be known). The training algorithm used in the solution was the well known *back-propagation* algorithm. Briefly, this works by initialising link weights randomly, then iteratively following a process of running all training image encodings through the NN, comparing the NN results to their actual classes (expected results), taking the mean-squared error, and propagating the error backwards through the NN to change the link weights in such a way as to reduce the error. This process

is described as a training iteration, and the number of training iterations to perform is an important consideration. Performing too few iterations will result in both inaccuracy and imprecision in the approximation of the bounding function, whereas too many will result in the bounding function fitting the training data *exactly*, such that generality is lost. These problems are referred to as under-fitting and over-fitting, respectively. An additional parameter involved in finding the optimal balance between these extremes is the number of hidden layer neurons, q. Fewer hidden neurons allow for less precision and therefore less classification 'power', leading to under-fitting the bounds of a class, however, too many can result in an NN that over-fits the bounding function by being too precise, and additionally more neurons make the NN more computationally expensive to train and run. Additional minor parameters in the training process are the 'learning rate' and 'momentum' of the back propagation algorithm, which essentially control how quickly the NN learns the bounding function. Provided these are set to non-extreme values they have little effect in practice.

The solution first used a single NN which was trained by the process described above for each problem. In the extended solution, a NN optimised for each problem on the parameters discussed was used as the basis for the NN ensemble. The ensemble architecture used in the solution was perhaps the simplest, where the ensemble was a collection of clones of the optimised NN for each problem.

The ensemble training is essentially a repetition of the training process for each NN, with different data used to train each NN. This training method was based on the standard *bagging* (bootstrap aggregating) approach for ensemble training. NN ensembles are based on the principle that it is in practice very difficult to train a single NN to approximate a complex function such as the bounding function of a class of images. By combining the results from multiple NNs which are trained well (but not perfectly) it is possible to ignore an occasional bad result from an individual NN since it is likely that there will be a consensus towards the correct result for any given classification. The bagging training method used is very much in line with this principle, and involves training each NN in the ensemble on a random 90% sample of the training set.

By sampling the training set, it is hoped that 'bad' training images, for example images which have some feature which is very rare in the general population, will not be included in the sample for one or more NNs. If a 'good' image is removed from the sample, it is not significant since there are many similar good images in the sample, but cutting just one bad image from a sample can greatly reduce or remove the representation of its unusual features from the training data for that NN. Thus, the NN ensemble is more inclined to 'agree' on good results, with 'bad' results occasionally thrown up by individual NNs contested by other NNs trained on slightly different samples, so overall accuracy and generality can improve.

D. NN-Unit Testing

The final stage of the solution was to test the trained NN-unit using the separate test set of images from the image database. The test set did not overlap with either the set of face images used to compute the eigenfaces, or the set of images used to train the neural network, therefore each test image could be described as 'unknown' to the solution. If face images used in the eigenface computation had been used for testing, the accuracy of classification would have been inflated somewhat since these images could be extremely well represented by the eigenfaces. This is not a meaningful result however, since in the vast majority of applications of an FR solution the faces encountered will not have been used to compute the eigenfaces. For a similar reason, testing on images which have been used to train the NN-unit would produce meaninglessly inflated classification accuracy rates. The explicit separation of

training and test data was therefore designed to avoid trivially excellent classification results and introduce confidence in the generality of the classification results attained.

The testing of the NN-unit, whether a single NN or an NN ensemble, is very straightforward. For the single NN, the test image (which was encoded in *stage **B***) is simply given as input to the trained NN, and the output is interpreted to give a classification for the image. For example, if for face detection the NN has been trained to give an output of 1 for a face image and 0 for a non face image, an output of 0.9 would indicate a face image (setting the threshold at 0.5 for example).

The testing process for the NN ensemble repeats the above for each NN in the ensemble, and the binary output from each NN is used in a 'vote'. The 'winning' output is thus the output with the majority of votes, which can be forced by using an odd number of NNs.

E. *Implementation and Testing Issues*

The previous sections summed up the theory and specifics of the solution, but did not focus on any particular implementation issues. Though the solution is assumed to be 'offline' and thus the speed of training and testing are not considered, the solution was implemented with a view to minimising running time and maximising flexibility, allowing for more results to be obtained for a richer evaluation of the solution.

As mentioned, the image database was managed using a local MySQL installation. The alternative would have been to use directories of files. However, using a MySQL database was far more convenient for managing the images, and allowed simple persistence of data from the training section such as the eigenfaces, image encodings, and trained NNs, to avoid having to compute this data again every time the solution was tested.

The solution was implemented solely using Java. The main reason for this choice was the various libraries available for the algorithms and tools needed for the solution whose implementation was beyond the scope of this project. These included eigenvector computation (JAMA) and neural networks (ENCOG). Similarly, the Java imaging and MySQL database connectivity libraries allowed for simple I/O of data and images.

The solution used a graphical user interface (GUI). This was partly for convenience in the administrative tasks such as organising the image database or changing NN parameters, but also helped in testing the solution implementation through the use of visualisations of the data. The nature of the solution made testing an inherently difficult task. Firstly, the eigenface encoding method analyses images in a non-intuitive manner through statistical analysis. This means that the eigenfaces and the image encodings are just vectors of values which mean nothing to the human observer. However, both an eigenvector and therefore an image encoding (which is a combination of eigenvectors) can be visualised as an image. By verifying that an image encoding 'looked similar' to the original, it was possible to deduce that the eigenfaces had been correctly computed and the original image had been successfully encoded as a combination of the eigenfaces.

The testing of the NN(s) was also difficult, since an incorrect result does not necessarily indicate an implementation problem, but could just be the result of poor training. Fortunately, the use of an NN library meant that the NNs could be assumed to be working correctly internally, but it was still necessary to test their application. In order to do this, a facility in the GUI for 'single' tests was set up, whereby the NN-unit can be trained on any parameters and then tested on any image in the test set. Thus, it was possible to test the NN-unit on arbitrary parameters and test images, allowing for directed investigations when an error was suspected in overall results. In addition, the error after each training iteration of the NN(s) in the NN-unit was output to the GUI, so that it could be seen how well an NN was learning and

at what rate. This allowed for testing of the training procedure, as this error would change unpredictably or be unreasonably high or low if there was an error with the training data, such as non-face images given to train an NN for gender classification or similar.

IV. Results

In this section results are presented, firstly from experiments on optimising the accuracy of the solution for each of face detection, face identification, and gender classification, and secondly from experiments testing the robustness of the general solution to noise and blur in test images.

A. Accuracy Optimisation Experiments

The first set of experiments involved optimising the general solution for each problem. The accuracy rate in terms of correct classifications was the overriding factor, but obviously there was a secondary consideration of the practical running time in determining the optimal configuration of parameters (see Table 1) for each problem.

In order to make the results representative of performance across the image database, the accuracy rates presented are the mean of 9 results obtained from running the solution 3 times on 3 different banks of training and testing images. To make the results non-trivial, the training images were kept as separate as possible from test images. Clearly, no test images were used in eigenface computation or NN training, but also (except for face identification) were not from individuals who had been involved in training.

The results for each problem follow a standard format. Firstly, the configuration of the image banks used is described. Then the results of optimising the single NN to achieve the highest possible accuracy rate are given. The optimisation was performed by altering each of K, q, and t, in order (see Table 1), carrying forward the optimal values found. When a parameter was altered, the others were frozen to keep the base 'power' of the NN constant (the only exception was when altering K, q was set equal to K to give the NN a power proportional the size of input). The accuracy rates of an ensemble of NNs using the optimal parameters are finally presented, varying the number of NNs used in the ensemble.

TABLE 1. SOLUTION PARAMETERS

Parameter	Symbol
Number of Eigenfaces Used in Image Encoding	K
Number of NN Hidden Layer Neurons	q
Number of NN Training Iterations	t
Number of NNs in NN ensemble	α

i. Face Detection

Bank configuration: Of the 40 individuals, a random 20 were in the training set, and the remaining 20 in the test set. 200 non-face images were placed in the training set, a separate 200 in the test set. Eigenface computation was done using all 200 faces in the training set.

Figures 9 to 11 show the single NN optimisation for face detection. The optimal K was 100. Using K = 100, the optimal p was 150. Using p = 150, the optimal t was 130. Table 2 shows the accuracy rates achieved with an ensemble of different numbers of NNs using these optimal parameters. With the single NN the maximum accuracy rate achieved was 97.83%, with the NN ensemble the maximum accuracy rate was an improved 98.08%.

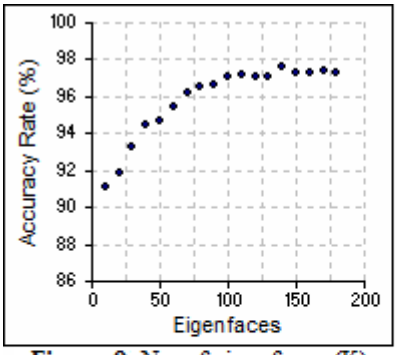


Figure 9. No. of eigenfaces (K) vs. accuracy rate for p = K , t = 100

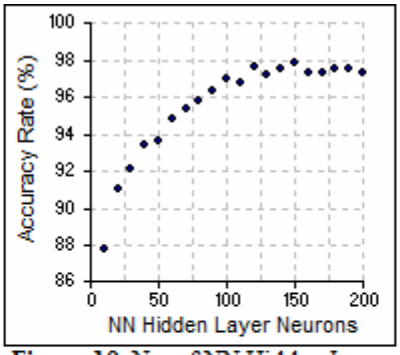


Figure 10. No. of NN Hidden Layer Neurons (p) vs. accuracy rate for K = **100**, t = 100

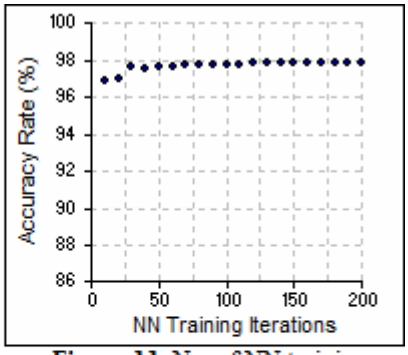


Figure 11. No. of NN training iterations (t) vs. accuracy rate for K = **100**, p = **150**

TABLE 2. ACCURACY RATES WHEN EXTENDING FROM SINGLE NN TO ENSEMBLE

Number of NNs in Ensemble, α	1	3	5	7
Mean Accuracy Rate (%)	**97.83**	**97.58**	**97.97**	**98.08**

ii. Face Identification

The results presented here are from face identification of one randomly selected individual, which can represent the general performance of the solution on this problem.

Bank configuration: Of the 10 images of the individual being identified, a random 5 are used for training and the remaining 5 for testing. Of the 39 other individuals, the eigenfaces are computed from the images of a randomly selected 20. From the remaining 19, 15 individuals are randomly selected, and one image per individual is randomly selected. 5 are used for training, and 10 are used for testing.

Figures 12 to 14 show the single NN optimisation for face identification. The optimal K was 80. Using K = 80, the optimal p was 130. Using p = 130, the optimal t was 50. Table 3 shows the accuracy rates achieved with an ensemble of different numbers of NNs using these optimal parameters. With the single NN the maximum accuracy rate achieved was 94.81%, and this was not improved upon (or even matched) by an NN ensemble.

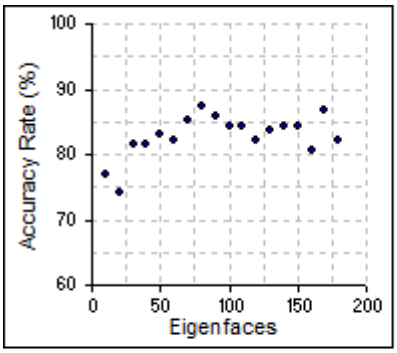


Figure 12. No. of eigenfaces (K) vs. accuracy rate for p = K , t = 100

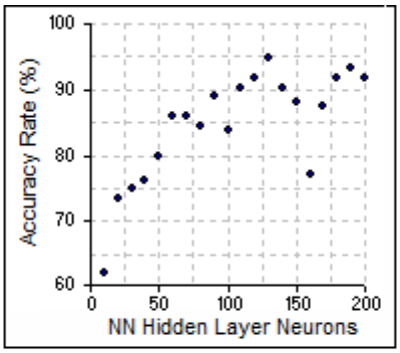


Figure 13. No. of NN Hidden Layer Neurons (p) vs. accuracy rate for K = **80** , t = 100

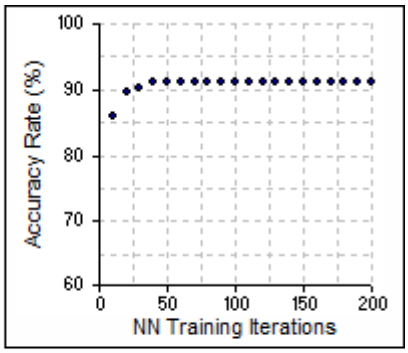


Figure 14. No. of NN training iterations (t) vs. accuracy rate for K = **80** , p = **130**

TABLE 3. ACCURACY RATES WHEN EXTENDING FROM SINGLE NN TO ENSEMBLE

Number of NNs in Ensemble, α	1	3	5	7
Mean Accuracy Rate (%)	**94.81**	**91.85**	**91.11**	**93.33**

iii. Gender Classification

Bank Configuration: 3 of the 4 female individuals, along with 3 males, made up the training set. The remaining female and 10 randomly selected male images were used for testing.

As will be explained in the next section, the results shown in Figures 15 to 17 and Table 4 were inconclusive. Though an attempt was made at optimisation, no meaningful improvement in results was seen.

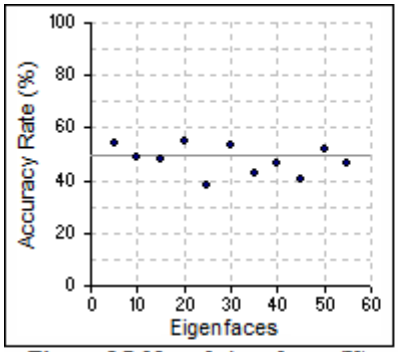


Figure 15. No. of eigenfaces (K) vs. accuracy rate for p = K , t = 100

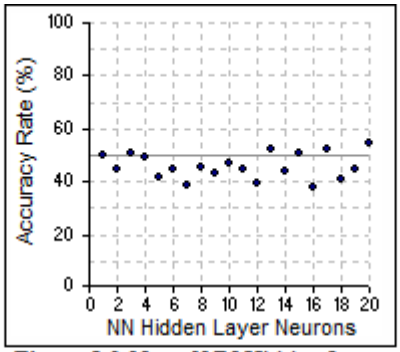


Figure 16. No. of NN Hidden Layer Neurons (p) vs. accuracy rate for K = **5** , t = 100

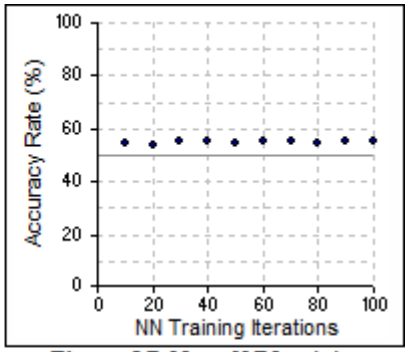


Figure 17. No. of NN training iterations (t) vs. accuracy rate for K = **5** , p = **13**

TABLE 4. ACCURACY RATES WHEN EXTENDING FROM SINGLE NN TO ENSEMBLE

Number of NNs in Ensemble, α	1	3	5	7
Mean Accuracy Rate (%)	**55.56**	**44.44**	**54.44**	**51.11**

B. Robustness Experiments

The final set of experiments tested the robustness of the optimised solutions to test images with artificially added blur and noise, respectively. The tests were performed by applying the optimal NN ensemble for face detection using the same image banks, except with blur and noise added to the test set.

Figure 18. From left to right, blur and then noise added to an image by 0,10,25,and 200 filter iterations

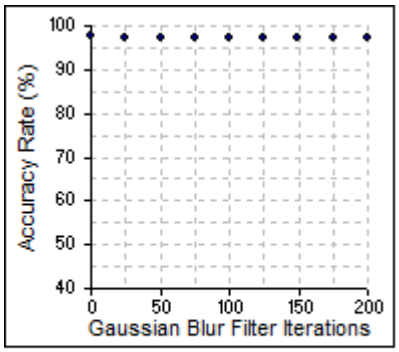


Figure 19. Increase in test image blur vs. accuracy rate for face detection

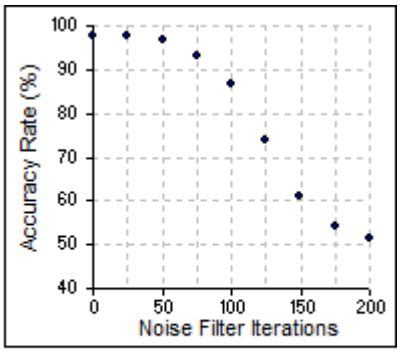


Figure 20. Increase in test image noise vs. accuracy rate for face detection

As can be seen in Figures 19 and 20, the accuracy rate was barely affected by increase in Gaussian blur in test images. As noise increased, however, the accuracy did fall until the solution did not work, yet the degradation in accuracy rate was graceful, as seen from the curve in figure 20.

V. Evaluation

In this section an evaluation of the solution is given using the results presented in the previous section. The evaluation will be done in terms of the questions posed in the first section, providing explanations of the successes and weaknesses the solution showed.

A. Application to and Optimisation for Each Problem

Figures 9 to 17 show the accuracy rates obtained when attempting to optimise the solution for face detection (FD), face identification (FI), and gender classification (GC). Overall, it can be seen that the solution was successfully optimised for the first two problems, but was not successful in its application to GC.

The results for both FD and FI show some of the strengths of the solution very well. In both problems, the high maximum accuracy rates achieved (98.08% for FD, 94.81% for FI) are of course relative to the images used, but are nonetheless impressive. Both problems were very responsive to optimisation, indicating two important benefits of the solution. Firstly, it shows that the eigenface encoding method was able to represent images for both FD and FI in a way which simplified classification. For FD this is more trivial, as clearly when encoding images using the principal components of face images, there will be an obvious consistency to those images which are faces. However for FI it also shows the ability of eigenface encoding to discriminate very well between individuals. Secondly, the advantages of NN classification were shown. In figure 10, the principles of under- and over-fitting can be seen, indicating that the NN was able to approximate the face class bounding function with such precision that it was too precise after an optimal 150 hidden layer neurons. In addition, the general response of both FD and FI accuracy rates to NN optimisation exhibits the ability of NNs to be easily optimised by altering just a few parameters. This is a great advantage over an algorithmic alternative, which can only improve accuracy through re-design.

The results for GC, however, are very poor indeed. As indicated in Figures 15 to 17 with horizontal lines, every result produced by the solution for GC was around the 50% accuracy rate. In a binary classification, this indicates that the results are meaningless, and just the mean of a spread of ‘random’ individual results. An attempt at optimisation was made, however there was no response. From the success of the FD and FI results, and the background theory, it appears that this failure was due to a flaw in the eigenface encoding. As previously mentioned, the eigenface encoding method emphasises the greatest differences between images, however in a problem such as gender classification, the differences required for discrimination may be more subtle than can be represented in a reasonable number of eigenfaces. In this regard, as previously mentioned, an alternative encoding method which may have been more appropriate is the Fisherface method. Of course, the number of female images available in the image database for eigenface computation was low, which only compounded this difficulty since even ‘global’ gender features were unlikely to have been captured.

B. Robustness to Image Noise and Blur

Figures 19 and 20 show that the solution was robust to added blur and noise, respectively, in test images. This was particularly the case for image blur, as indicated in figure 19, where the blur had a very minor effect on the accuracy rate. For test image noise, the accuracy rate degraded very gracefully indeed, with visually significant noise (see Figure 18) after 25 iterations causing only a very slight drop in accuracy rate, and the accuracy rate remaining acceptable until around 150 iterations, at which point the image is so heavily obscured that most information is lost, even to a human observer. The robustness of the eigenface decomposition encoding to both blur and noise is inherent in the use of eigenfaces. Although image blur and noise may appear very obscuring to the image as a whole, they do not significantly affect intensity variance on a pixel-to-pixel level. Therefore, in terms of their principal components (eigenfaces), which are the directions of greatest variance in pixel intensity with each pixel as a dimension, they are not dissimilar from their originals. Thus, image blur which is an averaging, and noise (in reasonable amounts) which is a random altering of pixel intensities, do not greatly affect the eigenface decomposition of the image.

C. Benefits of Neural Network Ensembles

Surprisingly, as shown in Tables 2 and 3 (and trivially in Table 4), the extension to NN ensembles was not as successful as had been hoped. In FD, a maximum accuracy gain of 0.25% was achieved using an ensemble of 7 of the optimised NNs, but this is not hugely significant, and in fact for FI there was no accuracy gain from the extension to NN ensembles. The simplistic training and voting methods used may have stifled the potential of the ensembles, and other factors such as varying the configurations of the individual NNs, or simply using more, would possibly have yielded higher accuracy gains. Regardless, in absolute terms it can be said that the NN ensembles did improve on the maximum accuracy for FD.

D. Discussion of Algorithms Used

To summarise, the techniques used in the solution performed well. It is difficult to see to what extent alternative encoding or classification methods could have performed better in FD or FI, although it must be said that the images used failed to 'stretch' these methods by introducing more unpredictable variation in lighting and background, for example. GC remains in a class of problem for which a reasonably accurate solution is out of reach using any current FR solution. The theoretical advantages of both eigenface decomposition and NNs were confirmed by practical success, although the success of NN ensembles was limited.

VI. Conclusion

In terms of fulfilling its objectives, the project was an overall success. A general FR solution using eigenface decomposition encoding prior to NN-based classification was implemented, and applied with great success to the problems of face detection and face identification on a non-trivial set of face and non-face images. The extension of the solution to using NN ensembles yielded some accuracy gain in face detection, but was otherwise unsuccessful in raising the maximum accuracy of the solution in face identification and gender classification. The gender classification problem was not solved by the solution on the images used, though

it is arguable that there would have been more meaningful results with a more appropriate set of face images incorporating a greater variety of female faces.

Therefore, the main findings of this project are as follows: Eigenface decomposition and a single NN for classification can solve the problems of face detection and face identification with high accuracy on face images with physical variations across gender, ethnicity, hairstyle, facial hair, and the wearing of glasses, and non-trivial variations of face rotation, position and size. The method does not perform well on gender classification, however. Furthermore, the solution is robust to blur and noise in input images, with the former barely affecting performance and the latter degrading performance gracefully for face detection.

There is a vast scope for future work which could extend this project. Perhaps the two most interesting avenues are more advanced NN ensembles configurations, and the combination of complimentary encoding techniques. These could be combined, by training each NN in an ensemble using a different image encoding technique, and then weight the voting depending on the type of problem, in order to optimise the advantages and disadvantages of each technique in one aggregate solution. Another path of investigation to improve the solution for gender classification would be to take two set of male-only and female-only eigenfaces, then project a face image to both eigenface spaces. By solving face detection in both spaces, the 'closest match' could indicate the gender of the face.

REFERENCES

[1] W.W. Bledsoe (1966), "The Model Method in Facial Recognition", Panoramic Research Inc., Palo Alto, CA, unpublished.

[2] T. Kanade (1973) "Picture Processing System by Computer Complex and Recognition of Human Faces", Dept. Information Science, Kyoto University, unpublished.

[3] S. Stillman, R. Tanawongsuwan, and I. Essa (1999). "A system for tracking and recognizing multiple people with multiple cameras." In *proc. of international conference on audio- and video-based person authentication*,pp. 96-101

[4] A.J. Howell, H. Buxton (1996), "Towards unconstrained face recognition from image sequences", *in proc. of 2nd international conference on automatic face and gesture recognition*, pp-224.

[5] J. Wilder, P.J. Phillips, C.H. Jiang, and S. Wiener (1996), "Comparison of Visible and Infra-Red Imagery for Face Recognition", *in proc. international conference on automatic face and gesture recognition*, pp182-187

[6] G. Gordon (1991), "Face Recognition Based on Depth Maps and Surface Curvature", *in proc. of SPIE, Vol. 1570: Geometric Methods in Computer Vision*, pp 234-247.

[7] L. Wiskott, J.-M. Fellous, N. Krueuger, C. von der Malsburg (1999) "Face Recognition by Elastic Bunch Graph Matching*", Intelligent Biometric Techniques in Fingerprint and Face Recognition*, pp. 355-396

[8] M Burton, V Bruce and N Dench (1993) "What's the difference between men and women? Evidence from facial measurement". *Perception*, 22, 153-176

[9] S Carey and R Diamond (1977) "From piecemeal to configurational representation of faces", *Science*, Vol. 195, P 312-13.

[10] L. Sirovich and M Kirby (1987) "Low dimensional procedure for characterization of human faces", *Journal of the Optical Society of America A*, 4(3), pp. 519-524

[11] M. A. Turk and A. P. Pentland (1991) "Face recognition using eigenfaces," *Proceedings of IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, pp. 586-591.

[12] M. Turk and A. Pentland (1991) "Eigenfaces for Recognition", *Journal of Cognitive Neuroscience*, 3(1), pp. 71-86.

[13] H. Abdi, D. Valentin, B. Edelman, (1998) "Eigenfeatures as intermediate level representations: the case for PCA models", *Brain and Behavioural Sciences*, Vol. 21, pp. 17-18

[14] P.N. Belhumeur, J.P. Hespanha, D.J. Kriegman (1996) "Eigenfaces vs. Fisherfaces: Recognition using Class Specific Linear Projection", *in proc of the 4th European Conference on Computer Vision*, pp. 45-58.

[15] Zhujie & Y.L. Yu (1994) "Face recognition with eigenfaces", *Proceeding of IEEE Conference on Industrial Technology,* pp. 434-438,

[16] L.K. Hansen and P. Salamon, “Neural Network Ensembles” *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(10), pp. 993-1001