

Lincoln University Digital Thesis

Copyright Statement

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

This thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- you will use the copy only for the purposes of research or private study
- you will recognise the author's right to be identified as the author of the thesis and due acknowledgement will be made to the author where appropriate
- you will obtain the author's permission before publishing any material from the thesis.

Using Mobile Devices to Access Remote Visualisations

A thesis
submitted in partial fulfilment
of the requirements for the Degree of
Master of Software and Information Technology

at
Lincoln University

by
Zitian Li

Lincoln University

2013

Abstract of a thesis submitted in partial fulfilment of the requirements for the
Degree of Master of Software and Information Technology.

Using Mobile Devices to Access Remote Visualisations

by

Zitian Li

Visualisation is widely used in many areas to help people interpret data. People may wish to share the visualisation that they are currently using with others. Mobile devices provide one way to share a visualisation, but they have limitations for visualisation compared to the use of standard desktop in terms of memory, graphics hardware and CPU performance.

In this thesis, a remote visualisation approach is used to overcome some of these limitations. As the Paraview visualisation system can operate in client-server mode, the focus is on investigating a solution that can efficiently transfer visualisations from the Paraview system onto a smart phone. The solution developed is an application that can view and update the rendered image generated from the existing visualisation system. The visualization system does not need to be re-built or installed on the smart phone. End users can easily retrieve the correct visualization without complex user configuration. In order to generate correct visualisation without having to re-build the client for the mobile device, the thesis uses a gateway approach acting as middleware to communicate between the render server (Paraview system) and our generic client (running on smart phone). The gateway handles all of the requests between the user and the server which runs locally to the gateway. The gateway passes the final rendered image to the smart phone via the network for display and interacting.

An end user trial is conducted to evaluate and compare the usability of the smart phone client with the traditional desktop client. The results showed that using gateway as a middleware is a useful approach for a smart phone to manipulate the visualisation with the Paraview system. The customised visualisation application on the smart phone is simple, without having to store data and without installing Paraview system on the device.

Keywords: Visualisation, gateway, client-server architecture, Paraview system, mobile client

Acknowledgements

It finally comes the time to write my acknowledgement page after countless nights that struggled with the writing. I would like to first express my deep gratitude to my supervisors, Stuart Charters and Keith Unsworth, who gave me constant help and guidance. This thesis would not have been possible without the support and assistance of them.

Secondly, I would like to express my gratitude to other professors and teachers at the Applied Computing Department in Lincoln for the friendly studying environment. Thank the postgraduates in Lincoln University, who helped me with my research for their enthusiastic participation.

I also own my sincere gratitude to my friends who offer me their helps in helping me work out my problems during the difficult time.

Last but not least, I would like to thank my beloved family for their encouragement and support through these years.

Table of Contents

Abstract	1
Acknowledgements	2
Table of Contents	3
List of Tables	5
List of Figures	6
Chapter 1 Introduction	7
1.1 Objectives	10
1.2 Structures of thesis	10
Chapter 2 Literature Review	11
2.1 Steps of Visualisation	11
2.2 Mobile device trends	11
2.3 Collaborative Visualisation.....	13
2.4 Distributed Visualisation	15
2.5 Mobile Visualisation in use	24
2.6 Summary	26
Chapter 3 Analysis and Design	28
3.1 Requirements.....	28
3.2 Objectives of thesis	30
3.3 Analysis	31
3.4 Design.....	32
3.4.1 Direct communication approach	32
3.4.2 Gateway approach	33
3.4.3 Selected approach.....	33
3.5 System Design	34
3.6 System architecture	35
3.6.1 Server	36
3.6.2 Gateway	37
3.6.3 Mobile client	39
3.6.4 Communication between server-gateway & gateway-client	41
3.6.5 Functions on mobile device	43
3.7 User perspective of application flow	44
3.8 Summary	45
Chapter 4 Implementation	46
4.1 System overview	46
4.2 System environment.....	46
4.3 Programming language.....	47
4.4 Gateway implementation	47
4.4.1 Sever-Gateway implementation	48
4.4.2 Gateway-Client implementation.....	49

4.5	Mobile Client Application Implementation	51
4.5.1	Mobile client user interface	51
4.5.2	Update functions.....	53
4.6	Network connection implementation	59
4.7	Summary	61
Chapter 5 Evaluation		63
5.1	End user trial	63
5.1.1	User trial environment.....	64
5.1.2	User trial procedure	64
5.2	Results & discussion.....	66
5.2.1	Observation.....	66
5.2.2	Completion Time.....	68
5.2.3	Questionnaire results and discussion	72
5.2.4	Interview results and discussion	78
5.2.5	Evaluation objectives: results and discussion	81
5.3	Summary	83
Chapter 6 Conclusion and future work		84
6.1	Conclusion.....	84
6.2	Future work and development	87
6.3	Summary	89
Appendix A User Trial Mobile Task		90
A.1	Mobile Task	90
Appendix B User Trial Desktop Task		91
B.1	Desktop tasks:	91
Appendix C User Trial Questionnaire		92
C.1	Interview Questionnaire	92
Appendix D User Trial Introduction Session		97
D.1	Instructions to participants.....	97
D.2	Mobile phone and web browsing familiarization	98
D.3	Demonstration	98
D.4	Functions on Mobile	99
D.5	Desktop Functions	102
References		104

List of Tables

Table 2-1 HTC phone & Computer Hardware comparison	24
Table 5-1 User trial environment	64
Table 5-2 Average time spent for each task.....	68

List of Figures

Figure 1-1 Construction model and mobile phone interaction (Sørensen K, 2009)	8
Figure 2- 1 Haber-McNabb Visualisation Pipeline (Haber & McNabb, 1990)	11
Figure 2- 2 eResearch Visualisation Scenario (Charters, 2010).....	13
Figure 2-3 Heer's Time-Space matrix (Heer, 2008)	14
Figure 2-4 Client-server approach (Garbow, Yuen, Erlebacher, Bollig, & Kadlec, 2003)	16
Figure 2-5 Dataflow in scientific visualisation applications (Luke & Hansen, 2002)	17
Figure 2-6 Design architecture and result (Lamberti et al., 2003)	18
Figure 2-7 RAVE architecture (Grimstead et al., 2005)	19
Figure 2-8 Interactions between client and server for the RAVE system (Grimstead et al., 2005)	20
Figure 2-9 RAVE service manager (Grimstead et al., 2005)	21
Figure 2-10 WEB-IS system architecture (Yuen et al., 2004)	22
Figure 2-11 WEB-IS system	23
Figure 2-12 Architecture overview (Diepstraten et al., 2004)	25
Figure 3-1 Classic Client-server Architecture	31
Figure 3-2 Direct Communication approach design.....	32
Figure 3-3 Gateway design.....	33
Figure 3-4 System architecture	35
Figure 3-5 Flow chart of system design.....	36
Figure 3-6 Server flow chart.....	37
Figure 3-7 Gateway flow chart.....	38
Figure 3-8 Mobile Client flow chart.....	40
Figure 3-9 Server- Gateway flow chart	41
Figure 3-10 Client - Gateway flow chart.....	42
Figure 3-11 User perspective of application flow	44
Figure 4-1 Gateway data workflow	47
Figure 4-2 Server - Gateway communication.....	48
Figure 4-3 Commands text file format	49
Figure 4-4 Connection Panel of Mobile client interface	51
Figure 4-5 Invalid input alert for gateway connection.....	52
Figure 4-6 Function Panel of Mobile client interface.....	53
Figure 4-7 Zoom function input	54
Figure 4-8 Rotation diagram	56
Figure 4-9 Rotation & Zoom visualisation	57
Figure 4- 10 "Wireframe" representation.....	58
Figure 4-11 Gateway – Mobile client network connection.....	60
Figure 4-12 Client infinite loop in python	61
Figure 5-1 (a) Average Completion Time for mobile tasks.....	69
Figure 5-1 (b) Mobile tasks first vs. mobile tasks second	70
Figure 5-1 Mobile tasks comparison figures	70
Figure 5-2 Average completion time for different tasks using Desktop	71
Figure 5-3 Smart phone usage among participants	72
Figure 5-4 Smart phone download file frequency	73
Figure 5-5 Easiness of completing Mobile Tasks.....	74

Chapter 1

Introduction

Visualisation is widely used in different areas such as medicine, architecture, geography, to help people interpret physical data and objects (Zhou, Qu, Wu, & Chan, 2006). For example, scientists use visualisation to interpret collected raw data such as air flow; engineers use visualisation to view the mechanical parts of engines to solve possible problems; doctors also use visualisation to check the functions of the human body.

Visualisation is defined as transforming data or information into visual pictures (Schroeder, Martin, & Lorensen, 2002). It is the process of exploring, transforming, and manipulating data to gain insight and understanding, which provides a way of seeing the unseen for scientists, physicians, and engineers.

There are different types of visualisation namely: scientific visualisation to help scientists transform data into scientific views; information visualisation to translate collected data to a visual form in order to help with decision-making processes. In terms of different types of displays: 2D visualisation represents a flat display of a scene while 3D visualisation shows a three-dimensional view.

Just as there are different types of visualisation, there are also various types of display device for presenting visualisations in different environments. The most common device is a standard monitor. Others include stereoscopic displays, Cave Automatic Virtual Environments (CAVE) and mobile devices, which are less common and used in particular circumstances due to their capabilities and size. However, since new technologies started to develop rapidly, mobile devices, such as Personal Digital Assistants (PDAs) and smart phones have become much more popular in our everyday life. Mobile devices act as smaller computers.

The performance of mobile devices is constantly improving. In particular, smart phones have more capabilities than just phoning, including collaborative applications that allow different users to communicate with each other in real time. In one example, based upon construction workers' and

architects' experiences, Sørensen, Christiansson & Svidt (2009) have developed a prototype system with Radio-frequency identification (RFID) to support construction processes at different project management levels.

(Figure 10 in web link: http://www.it.civil.aau.dk/it/reports/2009_06_itcon_ksb.pdf)

Figure 1-1 (a) Virtual model and physical components in construction (Sørensen et al., 2009)

(Figure 8 in web link: http://www.it.civil.aau.dk/it/reports/2009_06_itcon_ksb.pdf)

Figure 1-1 (b) Using mobile phone to interact with different resources (Sørensen K, 2009)

Figure 1-1 Construction model and mobile phone interaction (Sørensen K, 2009)

In Sørensen et al.'s project, a prototype was developed to support the working processes for real time project management, quality assurance and inventory management in order to provide further insight into the future potential and challenges of using mobile devices for viewing visualizations in real-time. Starting with a manual and paper-based checking and project follow-up process, the prototype in Figure 1-1 (a) illustrates how it can be done digitally, in terms of interacting with

different resources such as mobile phones and virtual models. It shows how different users can exchange their construction project information through different devices. Figure 1-1 (b) shows how people can exchange information efficiently using mobile phones while working on the construction site.

Although Sørensen et al. mainly focus on developing a conceptual prototype system for a construction company, it demonstrates that visualisation on mobile devices is becoming more and more important in current and future technology development.

In general, people use standard monitors (either a desktop or laptop monitor) to view and update visualisations. However, in certain situations, people may wish to share visualisations with co-workers in different geographical locations. For example, scientists may need to look at an updated visualisation while they are out of the office; or architects may need to retrieve and send amended drawings to other team members who are at construction areas; or engineers may wish to view complex mechanical parts while they are working outside the office. The mobility and size of mobile devices, such as smart phones, make them convenient devices to carry out such tasks.

Since a smart phone can actually act as a small computer, in addition to using a mobile device to present a scientific visualisation, people may also wish to interact with the visualisation.

In these scenarios, images are generated by an existing visualisation system. However, the transmission of the images to a mobile device, and interaction with the visualisation, may require installation of a visualisation system on the mobile device together with the development of an appropriate user interface. This may require all the limited resources of a mobile device and slow down the processing speed. Meanwhile, people may need to have knowledge about the complex visualisation system in order to interact with a visualisation. In addition a visualisation will need to re-scale to fit the screen of the mobile device for display.

1.1 Objectives

In this thesis, we describe the design and development of a solution to generate and subsequently update a scientific visualisation, using a mobile device, with an existing visualisation system. Such a system must be easy to use and not require users to work with a complex visualisation system.

The overall aim of the research is to provide this mobile device functionality, allowing users to interact and view a visualisation, on the mobile device through simple inputs using a customised visualisation application.

1.2 Structure of Thesis

Chapter 2 discusses related work in the areas of visualisation. It reviews the literature on the use of mobile devices in visualisation together with fundamental concepts of remote visualisation, summarising the limitations of using a mobile device. It compares different approaches when applying remote visualisation to a mobile device. Chapter 2 also suggests guidelines that are appropriate for mobile applications.

The analysis and design of the thesis is outlined in chapter 3. Implementation is discussed in chapter 4. The evaluation is reported in chapter 5 and chapter 6 concludes the thesis and includes a discussion on possible future work.

Chapter 2

Literature Review

In this chapter, we will introduce the steps in the visualisation process and discuss the increasing role of mobile devices in current and future technology. Then we will describe the use of collaborative visualisation for user interaction. In section 2.3, remote visualisation will be discussed in different real world scenarios. The limitation of using mobile devices in visualisation will be described in section 2.4. Finally, section 2.5 concludes by outlining an approach that is appropriate for mobile applications.

2.1 The Visualisation Process

Haber & McNabb (1990) describe the visualisation process in three major steps: filter, map and render as shown in Figure 2-1.

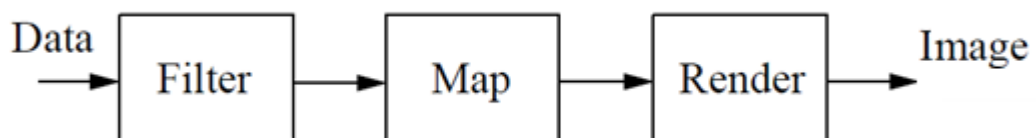


Figure 2- 1 Haber-McNabb Visualisation Pipeline (Haber & McNabb, 1990)

The pipeline illustrates how data are transformed into an image through those three steps. The quantity of data decreases as it moves from left to right in the pipeline. The filter stage is referred to as data enhancement. This first transformation deals with filtering the raw data and modifying the data for subsequent visualization operations. The second step maps the filtered data to a geometrical representation. Finally the rendering stage generates a visible image from the geometrical representation.

From raw data to finally present a visualisation, there are various techniques and platforms for generating a visualisation. Platforms for displaying visualisation have changed over time in recent decades.

2.2 Mobile device trends

Different mobile platforms have been developed to satisfy users' needs. Mobile platforms such as PDA, mobile and smart phones have changed rapidly in the last decade. PDAs (Personal digital assistants) are designed to help people dealing with electronic documents, such as reading emails. Smart phones combine the advantages of PDAs and mobile phones' calling capabilities, integrated

into a simple mobile device. In terms of visualisation, smart phones offer improved 3D rendering capabilities compared to a PDA.

Nowadays, mobile devices are playing an increasingly important role in visualisation. The motivation is that mobile devices and applications have been primarily designed to increase efficiency and productivity for people outside the office (Rodrigues, Barbosa, & Mendonça, 2006). Mobile devices such as PDAs and smart phones are widely used in modern society, and smart phones have seen very rapid technological development.

The performance of mobile devices is constantly improving. In particular, smart phones now have more capabilities than just phoning, including collaborative applications that allow different users to communicate with each other in real time, for instance, people now can easily post comments on the Facebook wall of other users through mobile devices.

Visualisation plays an increasingly important role in modern life. People use visualisation widely to analyse collected data and interpret trends in the real world. In order to satisfy different users' needs in terms of mobile device applications, necessary changes will need to be made to visualisation systems. Different devices will then be able to adopt these systems (Charters, 2010).

Figure2-2 shows a scenario for a scientist undertaking earthquake research while working at different locations.

It shows that people around the world may view results and analyse data in various locations and exchange feedback collaboratively. Display devices for visualisations are wide ranging, from a visualisation wall to small mobile devices such as a PDA or smart phone.

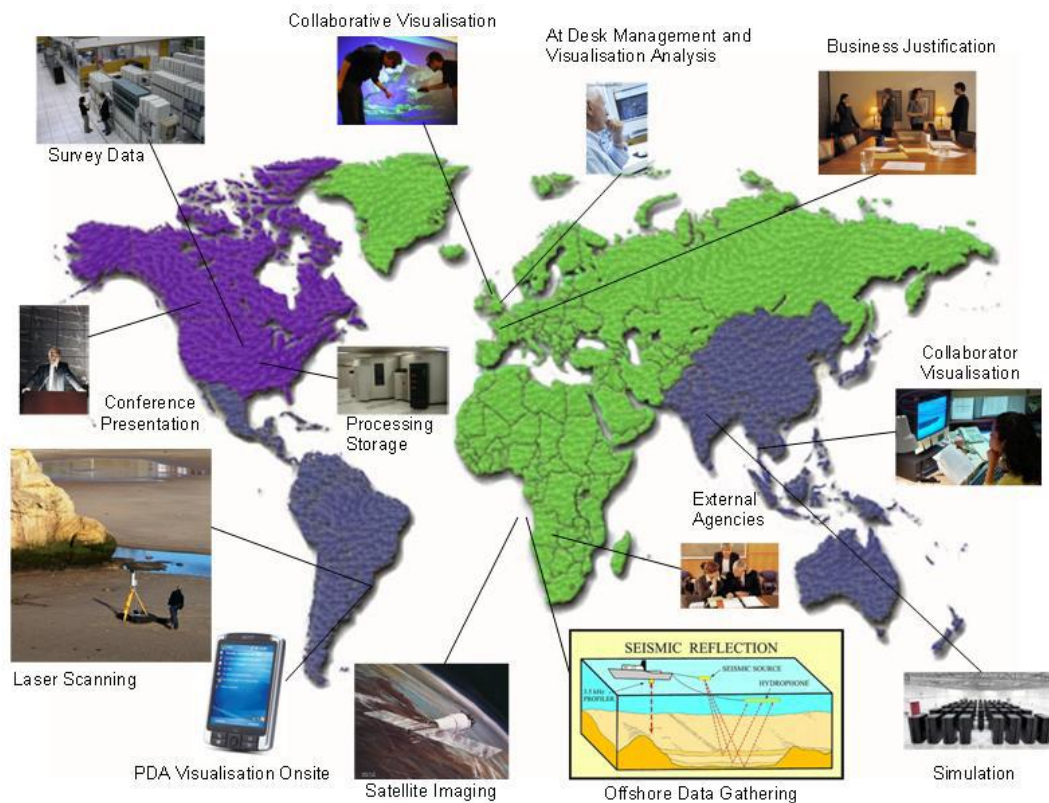


Figure 2- 2 eResearch Visualisation Scenario (Charters, 2010)

As visualisation systems and computing technology have developed significantly over the years, current visualisation systems need to be adapted and developed to support different devices in order to meet the needs of researchers in the future. Thinking of the rapid changes on mobile devices, especially smart phones, it is not surprising that there will be new demands of visualisation systems that are required to be supported on these mobile platforms.

2.3 Collaborative Visualisation

Data sets that are large may require more than one person to interpret them. Therefore such scientific research is carried out by teams rather than individuals. Collaborative systems allow multiple users to interact with the visualisation analysis process, for example, research scientists needing to look at a visualisation result, can contribute their ideas and share their interpretations with others in the team even if they are at different physical locations.

Collaborative visualisation enables different users to contribute their own understanding and views more quickly and easily to all members that are involved in the investigation. The fundamental approach to collaboration in computer systems is referred as computer-supported cooperative work

(CSCW). Applegate's CSCW time-place matrix (1991) outlines how people are involved in different activities based on time and place dimensions. Based on Applegate's CSCW matrix, Heer (2008) provides an updated time-space matrix for classifying collaborative applications. This is shown in Figure 2-3:

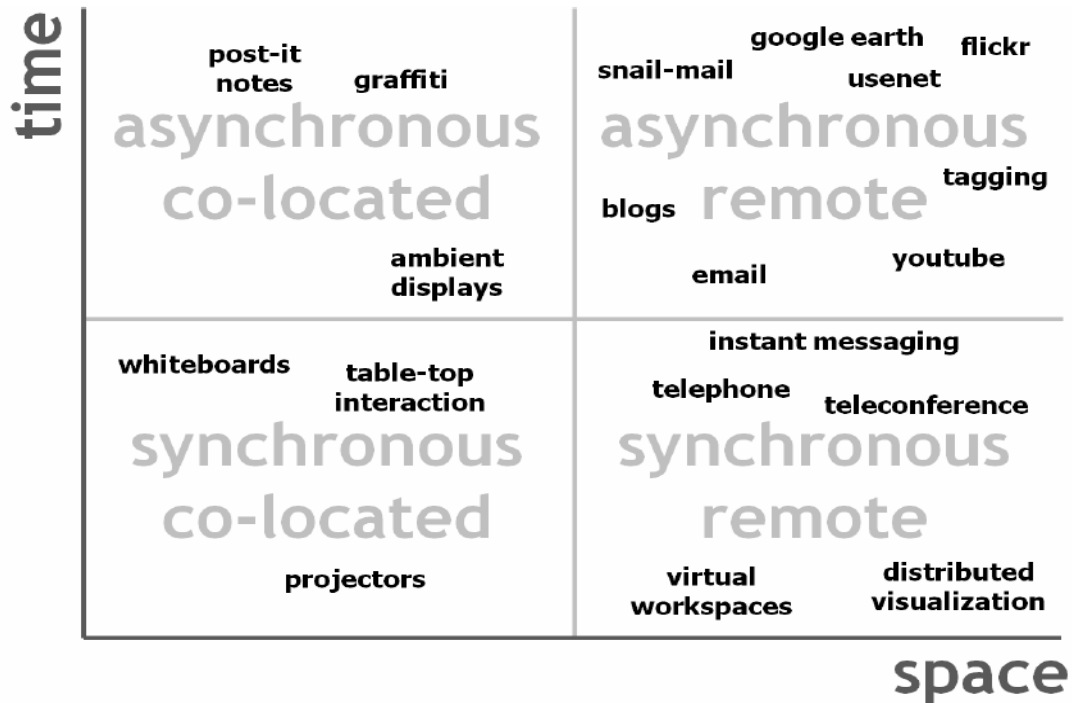


Figure 2-3 Heer's Time-Space matrix (Heer, 2008)

The Time-Space matrix also illustrates several technologies that have been applied for collaboration purposes. The time dimension indicates whether people interact at the same time or not (synchronously or asynchronously). The space dimension describes whether people are in the same (co-located) or different (remote) place. Brodlie, Duce, Gallop, Walton, & Wood (2004) also discuss asynchronous collaboration and synchronous collaboration. Asynchronous collaboration falls into different time dimensions. It refers to collaboration that involves activities such as letters, emails or faxes. Synchronous collaboration falls into the same time dimension. Activities such as video conferencing are classified as synchronous collaboration. In the synchronous remote dimension, distributed visualisation would allow people to share visualisations at the same time in different locations.

Pea (1993) explains that collaborative visualisation in the scientific area could allow scientists to establish real-time video connections with their colleagues, in the context of the sharing of computer

windows across networks in which scientific data and models can be discussed at a distance. It is known as “what you see is what I see”. The literature mentions that from the scientific point of view, the term “collaborative visualisation” refers to the development of scientific knowledge which is mediated by scientific visualisation tools in a collaborative context (Pea, 1993).

Other literature also states that one of the significant advantages of collaborative visualisation is that it allows for the pooling of remote computational resources (1999). Collaborative visualisation allows more users to take advantage of more advanced resources to aid the work process between members of the group.

2.4 Remote Visualisation

As mentioned earlier in section 2.1, in order to generate a visualisation, the necessary steps include data filtering, mapping and rendering (Haber & McNabb, 1990). There are many different computer rendering techniques for visualisation. Roberts (1993) states that isosurface and volume rendering are the main techniques for rendering volume data. Watt & Policarpo (1998) discuss many other important rendering techniques that have been developed, such as ray casting and ray tracing.

Depending on the rendering environment and the purpose of the visualisation, there are normally two ways for mapping graphics onto a device’s display: rendering the visualisation using a local application or through remote rendering of the visualisation (Rodrigues et al., 2006). Besides the rendering technique, user interaction is also another aspect that needs to be considered when displaying the visualisation.

Since researchers began to use more advanced and more powerful computers to process larger data sets, remote visualisation has become increasingly important (Luke & Hansen, 2002). There are situations in which scientists may need supercomputers or special graphics hardware to analyse data sets. However, those resources may be located in a different place to the scientist. In this case, remote visualisation is the solution to perform large scale visualisation using only limited resources.

Recalling the scenario described in chapter 1, people may like to view and interact with visualisation from a location outside their office. In this case, accessing the required visualisation with a device

which has limited resources may be an issue. One way to solve this problem is to use a remote visualisation capability. This allows users access resources from a remote system. This section describes some of the approaches to remote visualisation.

The fundamental components of remote visualisation normally include a high-performance graphics engine, such as those found in supercomputers or a desktop computer acting as a server; and another workstation in a remote location, such as other computers or mobile device, as a remote client.

(Figure 3 in web link: <http://www.msi.umn.edu/~lili/web-is.pdf>)

Figure 2-4 Client-server approach (Garbow, Yuen, Erlebacher, Bollig, & Kadlec, 2003)

The server and client communicate through a network. As shown in the example in Figure 2-4, this client-server approach uses a workstation as a server to perform some visualisation tasks while other multiple clients may at different locations interact with the visualisation remotely via a network using different display devices (Garbow et al., 2003).

Brodie et al.(2004) claim that since the early 1990s the client-server approach to visualisation has developed along with the Internet revolution. Client-server interaction varies depending on how much of the computation of the visualisation is done by the client and how much by the server.

The following figure illustrates different components of remote visualisation (Luke & Hansen, 2002):

(Figure 1 in web link: <http://dl.acm.org/citation.cfm?id=602107>)

Figure 2-5 Dataflow in scientific visualisation applications (Luke & Hansen, 2002)

Figure 2-5 shows different possible workloads for the server and client components of a visualisation system. Scenario 1 has the server performing most of the processing tasks, including data storage and rendering. The client only needs to display the image. In the second scenario, some of the rendering calculations are done on the server, the client finishes the rendering locally. In the third scenario, the server handles data storage and calculation, while the client performs all the rendering locally. The fourth scenario puts all the calculations and rendering on the client's shoulders and uses the server for raw data storage only, which is less common for remote visualisation.

Based on different remote visualisation approaches, Zhou et al. (2006) also describe three different modes of remote visualisation over networks:

- The simple solution for remote visualisation is a *thin-client mode* which means a light workload on the client side. This approach uses a high-end graphics server for complex computations, while a low-end graphics client, such as a mobile device, only needs to display the rendered image received from the server.
- The *fat client mode* is the opposite, which means most visualisation processing tasks are performed on the client side rather than the server side.
- The *balanced mode* divides the rendering and calculation processes between the client and server side, which sometimes is promoted as an ideal choice for mobile visualisations when the rendering tasks are not complex.

These three modes are similar to different components of remote visualisation (Luke & Hansen, 2002) in Figure 2-5. Depending upon the purpose of the visualisation, the hardware and software

performances of the mobile devices, we need to apply those three modes of visualisation appropriately according to different situations.

There are many different projects that use a client-server architecture to develop more advanced features. Lamberti, Zunino, Sanna, Fiume, & Maniezzo (2003) develop a framework that enables interactive remote visualisation of OpenGL GLUT-based graphics applications. See Figure 2-6.

(Figure 1 in web link:

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.1498&rep=rep1&type=pdf>)

Figure 2-6 (a) Client-server framework used in PDA

(Figure 5 in web link:

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.1498&rep=rep1&type=pdf>)

Figure 2-6 (b) Remote visualisation in Medical use

Figure 2-6 Design architecture and result (Lamberti et al., 2003)

Their concept is based on the Chromium architecture (Humphreys et al., 2002) to complete the rendering task with different workstation clusters. As the PDA has limited hardware to provide 3D rendering, each of the workstations renders a part of the image then sends it back to the PDA via a software bridge. The project also uses a client-server framework in the remote visualisation environment. The server and client exchange information over a socket through a network. The PDA device is used to control the remote rendering engine for collaborative work in the medical domain.

The user would be able to explore the visualisation interactively using a navigation interface. Compared with desktop devices, it emphasises the use of a server-based framework to generate large, complex and realistic 3D visualisations in order to overcome the PDA's limitations on high-quality 3D displays. This is a successful mobile remote 3D visualisation application but not available for general public use. It has been developed for the user who is familiar with OpenGL and is based on a PDA device.

The Resource-Aware Visualisation Environment (RAVE) system is a collaborative application that shares rendering resources between different platforms (Grimstead, Avis, & Walker, 2005). The RAVE architecture includes the following components: data service, render service, active render client and thin client. This is shown in Figure 2-7.

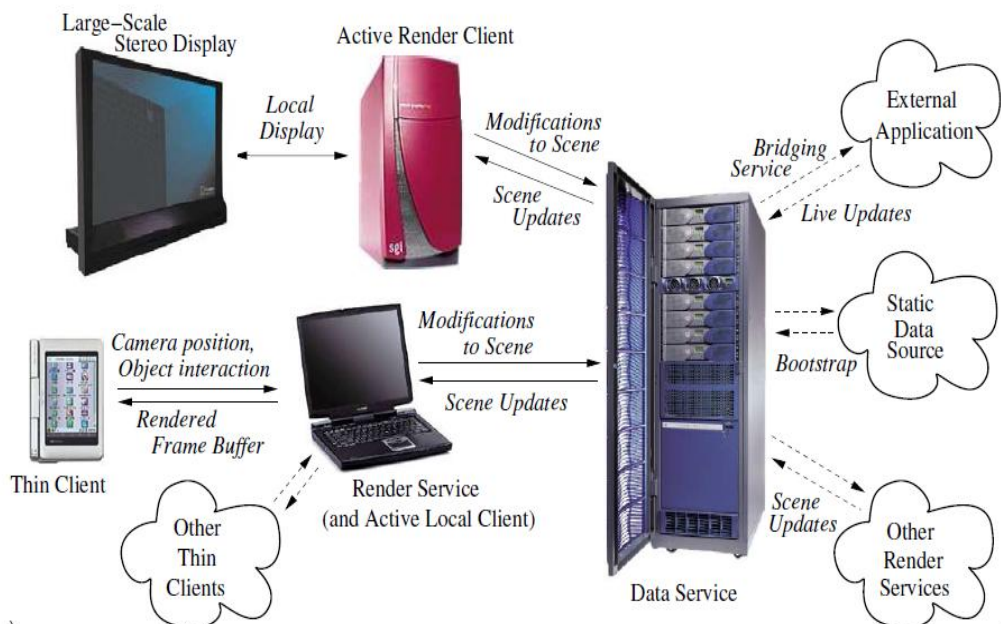


Figure 2-7 RAVE architecture (Grimstead et al., 2005)

The data service stores the rendering data locally and is responsible for recording any updates. The active render client represents a device that has a graphics processor and is capable of rendering the dataset, such as a laptop or desktop computer. The render service is similar to an active client except that it does not display the rendering results locally. The thin client is a client that has very limited rendering resources, for example, Personal Digital Assistants (PDA). Therefore, if the user has limited access to the rendering machine, the client could connect to the render server and request copies of the rendering results.

In the RAVE project, the PDA has insufficient internal memory to store the dataset and perform the rendering. It uses a render server to perform the rendering, a data server to store the dataset and a PDA to display the visualisation. The PDA user connects to the render server through a network and using a PDA application requests a copy of the rendering results. The render server confirms the connection with the PDA client and then sends a copy of the frame buffer over a TCP/IP socket to the PDA client.

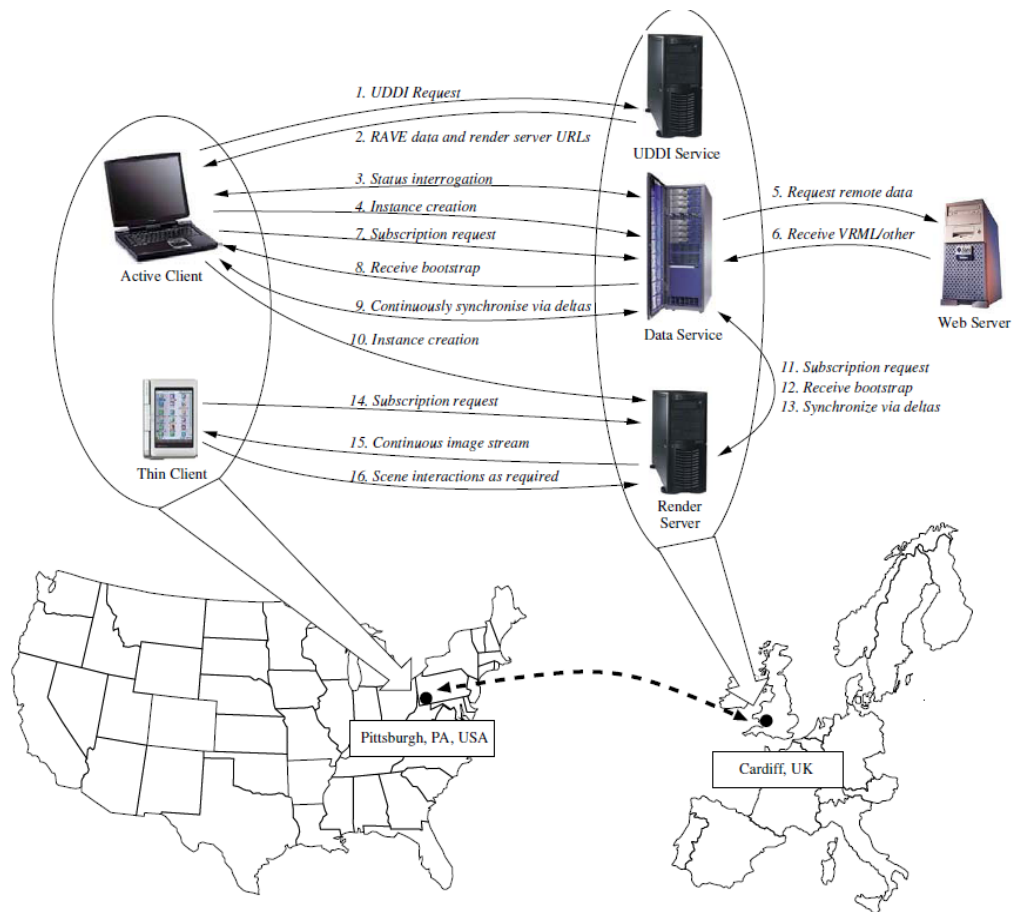


Figure 2-8 Interactions between client and server for the RAVE system (Grimstead et al., 2005)

In an experiment using RAVE, illustrated in Figure 2-8, remote visualisation was accomplished successfully using the client-server approach. The client and server were located in different countries. The client sent messages to request a TCP/IP socket to receive the data that the render server stored. After the render server received the message, it sent messages back to the client with the render server's port number and instance name created earlier. When the client received the information that the render server sent, the client connected to the render server's port. Once the

connection was established, the render server and the client continuously passed data over the network.

The screenshot shows the RAVE manager GUI with the following sections and annotations:

- Data Service Creation:**
 - (a) Service hosts discovered from UDDI, with statistics from interrogating each host. This points to the 'Available Data Services' table.
 - (b) Service instances hosted by selected service host, and (c) new instance creation. This points to the 'Data Service Instances' table and the 'Create new Data Service Instance' form.
- Render Service Creation:**
 - (d) Service hosts discovered from UDDI, with statistics from interrogating each host. This points to the 'Available Render Services' table.
 - (e) Service instances hosted by selected service host, and (f) new instance creation. This points to the 'Render Service Instances' table and the 'Create new Render Service Instance' form.
 - (g) Active and thin client creation, connecting to selected data or render service. This points to the 'Create new visualization client' section.

Figure 2-9 RAVE service manager (Grimstead et al., 2005)

Figure 2-9 shows the RAVE manager GUI, which is quite complex for users. Since the system is adopted across different platforms, there are many details that need to be specified in order to retrieve the visualisation output. In Figure 2-9 (g), the RAVE system will automatically generate the required rendered image from a web link on a thin client (PDA). Users can interactively choose different datasets and visualisation resolutions for display through this service manager.

The RAVE system provides the advantage of sharing resources between different platforms, particularly with regard to mobile devices, if the users need to generate visualisation results that require large amounts of computation.

In another remote visualisation application, Garbow et al. (2003) developed a web-based interrogative system that permits users to remotely analyse visualisation data based on a client-server framework. An advanced version of this web-based interrogative system called WEB-IS, enhanced by Yuen, Garbow, & Erlebacher (2004), extends the front end client using a wireless connection, such as a PDA. The advanced WEB-IS system is designed for geoscientists who want to

generate visualisations of large datasets for data analysis purpose. The components of the advanced WEB-IS system is schematically showing as Figure 2-10:

(Figure 2 in web link: <http://link.springer.com/article/10.1007%2Fs10069-003-0012-z?LI=true#>)

Figure 2-10 WEB-IS system architecture (Yuen et al., 2004)

The WEB-IS system allows collaborative visualisation. Clients can remotely retrieve rendered images using middleware called the web gateway. The web gateway offers different access points to collaborative clients. The WEB-IS system stores data and renders images off-screen on the server side, which enables multiple clients to render different data simultaneously. The server and clients use a web browser to exchange requests and update the images.

The web gateway is in the form of a web browser as shown in Figure 2-11 (a). It shows that WEB-IS provides a Java applet at the client end, which users can use to interactively update the visualisation results by selecting subsets of data to analyse.

(Figure 1 in web link: <http://link.springer.com/article/10.1007%2Fs10069-003-0012-z?LI=true#>)

Figure 2-11 (a) Web-based view via Java applets (Yuen et al., 2004)

(Figure 5 in web link:

<http://link.springer.com/article/10.1007%2Fs10069-003-0012-z?LI=true#>)

Figure 2-11 (b) WEB-IS system function setup (Yuen et al., 2004)

Figure 2-11 WEB-IS system

Figure 2-11(b) shows that WEB-IS uses a wide range of program tools such as C++, OpenGL, CORBA, Java to implement the server and client. Rendering on the server is generated by the Mesa3D graphics library in order to visualise the dataset. The http server is viewed as an applet from a web browser, which is part of the web gateway. Once the network connection is established, the client

makes requests to the server using a web browser and a Java applet as the front interface. The web gateway uses a CORBA bus to transmit commands and the image buffer to the server. The server will then pass the generated image back to the client for viewing via the web browser.

2.5 Mobile Visualisation in use

In comparison with mobile devices in the last decade, there have been significant changes in the use of mobile devices, especially mobile phones. However, besides the advantages of the mobility of mobile devices, there may be limitations in using mobile devices for visualisation compared to the use of standard computers, such as:

- Clarity of visualisation: small window size, lower resolution (Chittaro, 2006)
- Rendering speed: limited memory, graphics hardware (Chittaro, 2006); limited wireless network bandwidth (Park, Kim, & Ihm, 2008)
- Slower systems, restricted user interface, and lower processing performance (2006).

Table 2-1 outlines some hardware differences between a smart phone and a computer:

Table 2-1 HTC phone & Computer Hardware comparison

	HTC Touch Pro2 smart phone	General Computer
Core	Single	Dual Core
Processor	528 MHz	2.2 Ghz
RAM	288MB	2GB
External Storage	Up to 32GB	Up to 2TB
Screen (pixels)	480 X 800 (3.6'')	1400 X 900 (20'')
Graphic Performance	Dedicated graphics chip (64MB RAM reserved for graphics)	Choice between Dedicated or Integrated graphics chips
Network	HSDPA/WCDMA – From 9.6Kbps up to 9.2Mbps WiFi – Up to 54Mbps	Wired Network: Normally 100Mbps, can up to Gbps

The comparisons in Table 2-1 indicate that compared with a computer, current smart phones have lower graphics performance, slower processing speed and limited storage capacity. Most hardware features of the smart phone are similar to a late 1990s computer. However, Huang, Bue, Pattath, Ebert, & Thomas (2007) point out that there has been recent significant developments in computational power for mobile devices. Improvements continue to be made in processors, graphic

chips and display hardware; which means that mobile devices will become increasingly powerful compared with current specifications.

Chittaro (2006) also indicates the trend of displaying visualisations on small screen mobile devices. However, despite the significant advantage of the devices' mobility, Chittaro (2006) outlines the mobile devices' restrictions such as limited screen size, aspect ratio, hardware, connectivity and input techniques that may affect visualisation effectiveness.

One significant problem of mobile remote rendering is network bandwidth. Diepstraten, Görke, & Ertl (2004) address the issue of having low bandwidth compared to wired networks of desk computers when displaying 3D graphics on mobile devices, such as PDAs. The focus of their approach is to reduce the necessary network traffic when passing the rendered image to the PDA. The overall goal is to have a fair balance between client and server during the remote rendering process. Diepstraten et al. (2004) introduced a method to obtain a rendered image on a mobile device. The idea is to transfer just the 2D edge lines from the 3D image to the client device instead of the entire image.

(Figure 1 in web link:

<http://cumbia.visus.uni-stuttgart.de/ger/research/pub/pub2004/cgi04-diepstraten.pdf>)

Figure 2-12 Architecture overview (Diepstraten et al., 2004)

The system is shown in Figure 2-12. The thickness of the arrows represents the amount of data transferred over the network. The proposed solution was tested using different bandwidths. The results indicate that with different network settings, the client side frame rates did not seem to relate to the network bandwidth. Instead the frame rates depended on how fast the server could produce the necessary lines. This approach of changing the data that is transferred from images to lines, although not suitable in detailed visualisation environments, still gave a good hint on how to deal with the usual network bandwidth issue when using remote rendering.

In relation to the above literature, the bandwidth issue will need to be considered based on the server's rendering speed. Therefore, in order to overcome some physical limitations of mobile devices, such as limited memory capacity and lower graphics performance, remote rendering is the most common solution adopted in the literature for mobile devices (Nadalutti, Chittaro, & Buttussi, 2006).

According to Luke & Hansen (2002), as mobile devices have limited rendering resources, the scenario 1 dataflow in Figure 2-5 is most appropriate for mobile devices using a remote rendering approach, i.e., thin client mode using the mobile device as a display client and perform data processing and visualisation rendering on a different server.

2.6 Summary

Visualisation is the process of taking raw data and then filtering, mapping and rendering it. As raw data become larger, how to effectively use available resources to generate visualisations become more complex. Collaborative visualisation enables different users to contribute and share their own understanding and views more quickly and easily to all members that are involved in an investigation or research process. In particular, the synchronous remote configuration that allows people to exchange information at the same time in different locations is the focus of this project. This technique takes advantage of sharing more advanced resources to aid in the work process between members of the group.

Mobile devices such as PDAs and smart phones have become popular to use as a small computer. Increasingly users may want to view their visualisations on a mobile device. As mobile devices have limitations in processing power and graphic hardware, the client-server architecture approach to remote visualisation is an approach that may overcome some of the issues faced when dealing with the limited capabilities of mobile devices. Network bandwidth may be an issue if there is a slow rendering speed in a remote visualisation system.

However, the thin-client mode offers a solution to display the image only on the mobile device and use a high-end graphics server for the complex computations. Furthermore, some projects use web-

based applets to allow users to interact with the visualisation and choose appropriate datasets for updating visualisations.

The WEB-IS system and the RAVE system both produce visualisations via their own integrated programs and use a web server as middleware to allow the client to interactively communicate with the server. Both systems use a wide range of program tools to build the server, the middleware web server and the client; the systems program their own servers to analyse datasets in order to satisfy their particular needs, such as for geographic purposes in the WEB-IS system. The RAVE system has been developed for a particular PDA and requires visualisations to be created entirely using the RAVE system. On the other hand, both the client interfaces of those systems are complex for a general user to conduct remote rendering. The visualisation output is displayed through a web browser which may need to be reformatted for the user to view on a small mobile device such as a smart phone.

In order to overcome the disadvantages of these visualisation systems, a thin-client prototype design is introduced based on an existing visualisation system. The analysis and detailed implementation of this design is discussed in chapters 3 and 4.

Chapter 3

Analysis and Design

Chapter 1 describes the increasing use of mobile devices in current and future technology development and the limitations in using mobile devices for visualisation compared with desktop computers for example. Chapter 2 outlines some examples of using a client-server approach to overcome the limitations in mobile device (PDA) visualisation. In this chapter, we present an analysis of the requirements of this project and investigate a remote rendering solution to solve the problems raised in Chapter 1 and Chapter 2.

3.1 Requirements

An essential factor of the system is that the visualisation application should be able to help users view and update visualisations using a mobile device. The system should operate without having to store raw data or install a full visualisation system on the device. The application should be designed to provide the mobile device with functionality which is easy to use. The expectation therefore is that a visualisation will be delivered via a network to the mobile device.

We have determined the following requirements for the application:

1. Runs on a smart phone
2. Connects to a remote server
3. Interconnects with an existing visualisation system
4. Allows users to interact with the visualisation

1. Runs on a smart phone

In Chapter 2, we have included some examples of successfully using PDAs to display visualisations based on their visualisation system, such as the RAVE system.

For different mobile devices, their functions and capabilities are diverse. Compared with PDAs, a mobile phone such as a smart phone has a similar operation environment but different

functions and capabilities. In this research, as the visualisation application is aimed to help users view and update visualisations displayed on a mobile device, and since other projects have used a PDA as an example, and smart phones nowadays are capable of using different programming language to develop suitable applications, we use a smart phone as a mobile display device to run the visualisation application.

2. Connects to a remote server

As mentioned in Chapter 2, a remote rendering approach can overcome the shortcomings of smart phones.

After a visualisation has been generated, the solution should allow the smart phone to render the visualisation remotely by implementing a client-server architecture, i.e., the smart phone connects with a server remotely.

3. Interconnects with existing visualisation system

The main emphasis of the thesis is on developing a solution so that a smart phone can be used to view and update visualisations in remote locations. We are not concerned with developing a new visualisation system.

In addition, if the smart phone application is developed using an existing visualisation system, it is possible that the solution may work with other visualisation software. Therefore, it is necessary that the smart phone application interconnects with an existing visualisation system.

4. Allows users to interact with the visualisation

One of the main purposes of the application is to allow users to update a visualisation by using a smart phone. The solution should allow users to interact with the current visualisation using an existing visualisation system.

3.2 Objectives of Thesis

The overall objective of this thesis is to describe the development of a prototype that uses a smart phone to view and update a rendered image by interacting with an existing visualisation system. The design and development of the prototype includes several key objectives as follows:

1. Develop a solution that allows users to create a connection from the mobile device to the visualisation system in an easy way

Network configuration and interaction between server and mobile client can be complex and require knowledge of the rendering server and the network connection to the client. In order to reduce the complexity of communication between server and client, the solution should simplify this procedure as much as possible so that users can connect the smart phone with the remote server in a straightforward manner.

2. Develop a solution that allows users to interact with the visualisation system and update visualisations from a smart phone

One of the main objectives is to send the updated visualisation from the visualisation system server to the mobile client, such as a smart phone, based on the remote visualisation framework.

The solution should not only be able to send a single static image that represents the visualisation, but also be able to allow users to manipulate the received visualisation and update it by interacting with the visualisation system.

3. Develop a user friendly interface that can provide data validation and pass useful error messages back to the user

If end users input invalid data to the smart phone, the application should be able to automatically check the input data and generate meaningful error messages to alert user.

4. Develop a prototype that allows the solution to be deployed, using a similar visualisation system

The concept that will be developed should be applicable to similar existing visualisation systems. In principle, the communication protocol between client and server should be able to be used for more general use.

3.3 Analysis

As described in Chapter 1 and Chapter 2, remote visualisation offers a solution for overcoming some of the limitations of mobile devices. By using a client – server architecture, users can retrieve a visualisation for display on a mobile device using a higher-end workstation to process and calculate the complex data sets, without needing to use the mobile device’s limited resources to perform the visualisation.

Depending on the hardware and the purposes of the mobile client, the client – server architecture will differ depending on whether rendering occurs locally or remotely. The principle of the proposed client-server architecture of the existing visualisation system can be illustrated as follows.

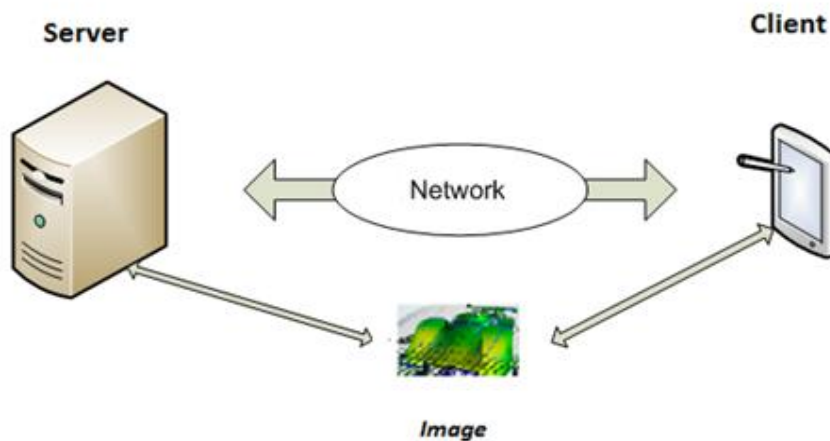


Figure 3-1 Classic Client-server Architecture

The client – server approach uses a thin client as the end user, i.e. PDA or smart phone, to display visualisations. The client and server communicate over a network. Visualisations are transmitted

between the two. Figure 3-1 shows that the computer acts as a server to perform data storage, filtering and mapping. Depending on the size of the dataset, data rendering may occur locally on the client or remotely on the server. Display of the visualisation happens on the client.

3.4 Design

We consider two approaches to the client – server architecture:

- Direct communication approach
- Gateway approach

These are discussed in more detail below.

3.4.1 Direct communication approach

The concept of direct communication is to render the visualisation on a smart phone directly through the server. Basically, both the server and the client will have to install visualisation application software in order to retrieve the visualisation using the smart phone as the application’s remote client. In this approach, we process the rendering window on the smart phone. The server will stream the data directly to the mobile client. This approach is illustrated in Figure 3-2:

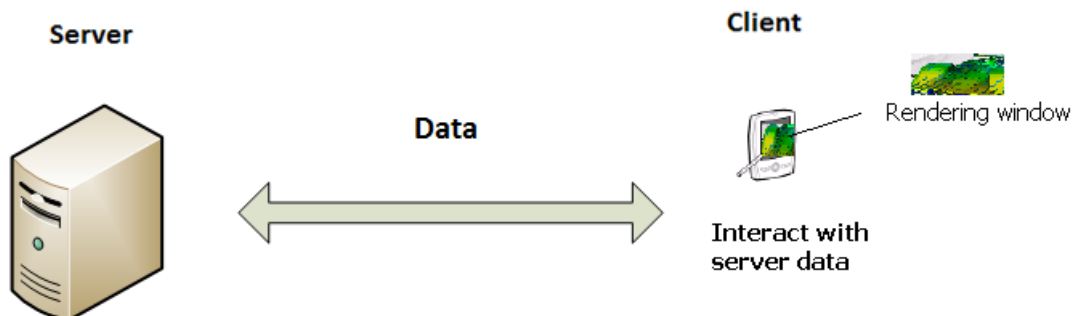


Figure 3-2 Direct Communication approach design

By using this approach, the mobile client will interact with the server directly. The mobile client does not hold a record of what and when an image has been sent from the server. Meanwhile, in order to render the image on the mobile client, it may be very complicated to build a version of the existing visualisation system on the mobile client. Also, we will have to consider the mobile client memory needed for successfully running the application client locally, since in order to run the application client, the library must be installed on it, which will take up the mobile client’s limited memory storage when installed. Once the data has been processed on the server, the visualisation will be

rendered on the phone using the visualisation software. Therefore, the library may also require a large amount of memory to run the application locally on the mobile client.

3.4.2 Gateway approach

The gateway approach originally came from the medical mobile collaborative display system of Park et al. (2008). However, instead of using a gateway to communicate with different collaborators, we can use a gateway to act as middleware to communicate between the server and client. In this approach, the smart phone is used for display purposes, after receiving the final rendered image. Also the user can use the smart phone to send update requests to the server via the gateway. This approach is shown in Figure 3-3:

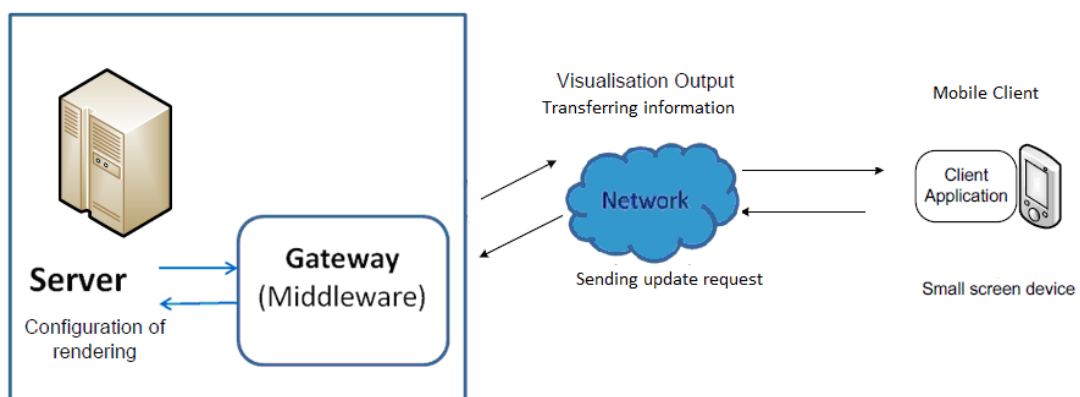


Figure 3-3 Gateway design

We run the application in client-server mode between the server and the gateway. The server completes the configuration for rendering, and streams the geometry data to the gateway. The gateway handles interactions locally, i.e., interacting with the server; receiving update requests from the mobile client and transferring those requests back to the server. We pass the visualisation output from the gateway to the mobile client through the network for subsequent display.

In this approach, the mobile client acts as an end user. It is responsible for displaying the received image and also allows the user to update the received image by interacting with the gateway.

3.4.3 Selected approach

Since both approaches are based on the remote visualisation architecture, we need to choose the appropriate one for our case. Instead of the server directly communicating with the mobile client,

the gateway is developed to control the interaction. In this thesis, the gateway design is selected due to the following advantages:

- Gateway controls what data or visualisation is sent to the mobile client

End users on the mobile client side do not need to know what happens between the server and the mobile client itself. The gateway is responsible for setting up the connection with the server and transmits the visualisation result to the mobile client. When the mobile client sends an update request on the client application to the server, the gateway will handle those requests and communicate with the server in order to get the updated image; when the updated image is ready, the gateway will notify the mobile client and send the image through the network.

- Do not need to install the visualisation application software on the mobile client.

In comparison with the direct communication approach, the gateway design can save the mobile client's memory space for running the visualisation application. In the gateway design, the server renders the visualisation; the gateway transfers requests and feedback while the mobile client is only used as a display device.

The mobile client does not have large memory space to store and render the visualisation, and this gateway approach allows the mobile client to access external resources, such as the visualisation system on the server, to render and store the image.

3.5 System Design

Figure 3-4 shows how the visualisation can be transferred between server and mobile client based upon the gateway design.

The system is based on three major components: server, gateway and mobile client.

These components are briefly outlined as follows. Detailed explanations are given in section 3.6.

- Server

It is responsible for data storage, rendering and updating the rendered image.

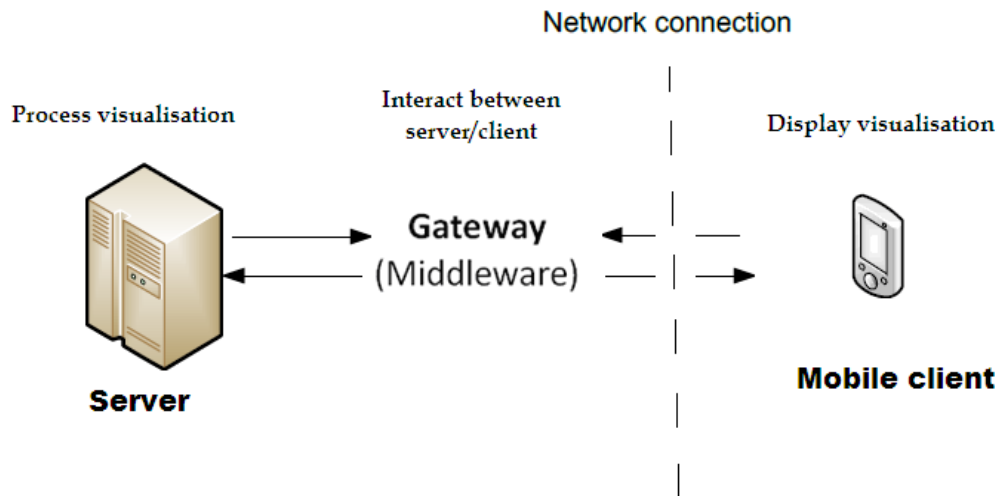


Figure 3-4 System architecture

➤ Gateway

The gateway and server are installed on the same computer. The gateway is designed to reduce computing work at the mobile client end. All communication between server and mobile client will be recorded and transferred via the gateway.

➤ Mobile client

It runs the visualisation application to connect with the gateway. A visualisation will be displayed via the application on the smart phone. The end user is able to use the smart phone to interact with the visualisation.

3.6 System architecture

In our project, client-server architecture is adopted to overcome some mobile devices' limitations for visualisation, i.e., limited memory storage, lower graphic performance. As described in the literature review section, the client-server thin client mode uses a high end workstation as a server to store data and perform rendering for the visualisation process while users at a different location can view the visualisation result remotely via a network using a smart phone.

Based on the original client-server framework, we have developed our solution which has a gateway between server and client to handle all the communication. The gateway is used to reduce the complexity of the user interface and control what is sent to the client. Figure 3-5 represents the data flow in the three major components:

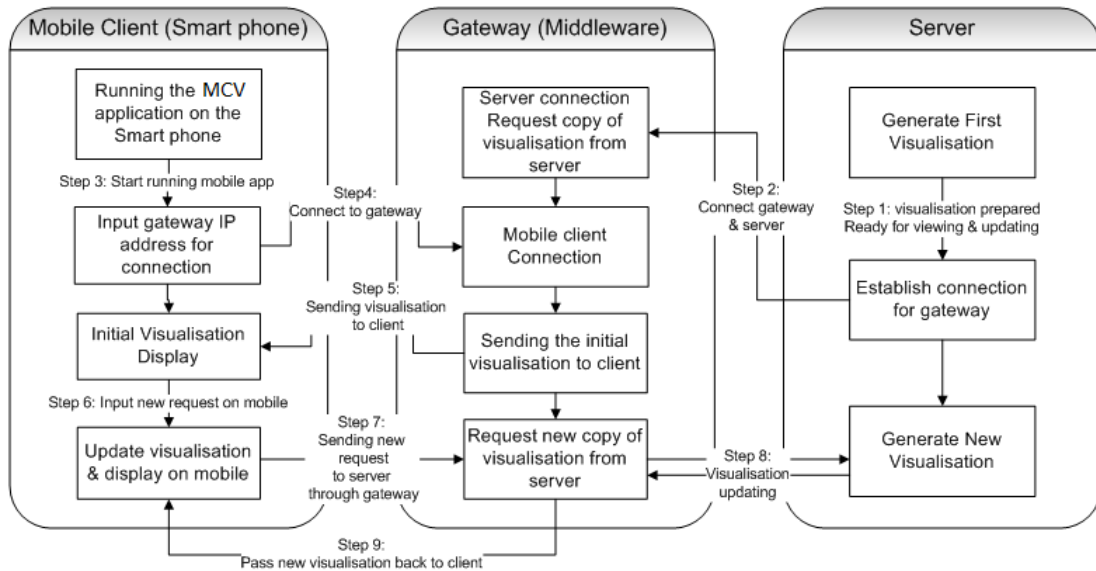


Figure 3-5 Flow chart of system design

It illustrates communication between the server, gateway and client. “MCV” in Figure 3-5 represents the mobile client visualisation (MCV) application that runs on a smart phone. The nine steps of the interaction will be described fully in the following sections.

3.6.1 Server

The server is based on the visualisation system’s built-in server. It is used for:

- Data storage, including uploading raw data
- Processing data or update requests and generating visualisations
- Sending visualisations to the client

When the server sends visualisations to the client, the ‘client’ in this case will be the gateway.

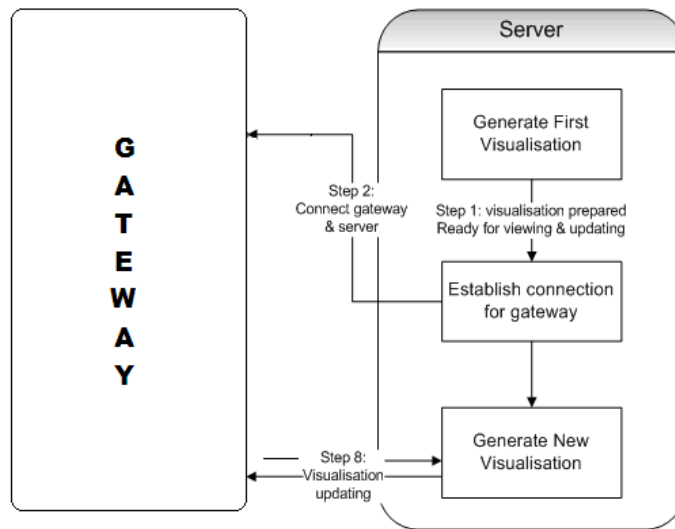


Figure 3-6 Server flow chart

Translating the proposed data flow shown in Figure 3-6, firstly, the server will upload data and generate the image locally through the existing visualisation system. In step 2, the server will open the connection, waiting for the gateway to connect with it. After the connection between server and gateway is established, the server will send the generated visualisation to the gateway.

The last process (step 8) is to generate a new visualisation. In this process, the server has already connected with the gateway. If the mobile client sends a request to update the visualisation, the gateway will transfer the request to the server through the connection. The server will then generate a new visualisation from the existing visualisation system and deliver the visualisation to the mobile client via the gateway.

3.6.2 Gateway

The gateway handles interactions between the server and the mobile client locally, such as receiving updated requests from the mobile client and transmitting commands back to the server. The gateway design flow is illustrated in Figure 3-7.

It shows that the gateway is used for completing four major processes:

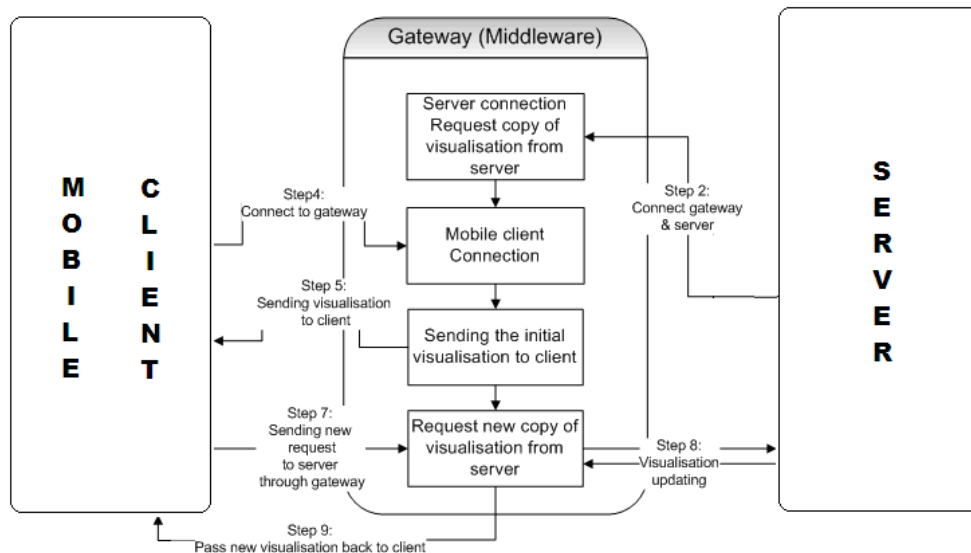


Figure 3-7 Gateway flow chart

1. Server connection

After the server generates the initial visualisation and opens the server for connection to the gateway (step 2), the gateway will need to establish a connection with the server in order to request a copy of the visualisation from the server.

2. Mobile client connection

Besides the connection between the server and the gateway, the gateway will also need to be able to communicate with the mobile client.

In step 4, the gateway will establish a connection between the mobile client and itself in order to handle interactions between the server and the client. The gateway will then use this mobile client connection to transmit visualisations generated by the server to the mobile client in steps 5 and 9; and pass requests from the mobile client back to the server in step 7.

3. Sending visualisations to client

Once the connection from the server to the mobile client through the gateway has been established, the server will transmit visualisations through the gateway to the mobile client.

During this process, the gateway requests a copy of visualisation from the server, and then sends the visualisations to the mobile client for display. See step 5.

4. Request new visualisation from server

The gateway is not only used for transmitting visualisations from server to client, but also can be used for exchanging requests between the two sides.

After the mobile client receives the initial visualisation from the gateway, the user may wish to view the visualisation differently by manipulating the mobile client display. If the end user sends a new command to update the current visualisation (step 7), the gateway will pass those update commands to the server in order to render the new visualisation remotely. The new visualisation is transmitted back to the mobile client via the gateway (steps 8, 9).

Based on the above four major processes, the gateway plays an essential role. It is specifically designed for managing data (including visualisations and update commands) between the server and the mobile client. The gateway takes responsibility for accessing datasets on the server and keeping track of the generated visualisations.

3.6.3 Mobile client

The mobile client is also called the end user client. The end user will use the MCV application to interact with the gateway. The basic logic of the mobile client is shown in Figure 3-8:

It shows the main procedures that the mobile client should have:

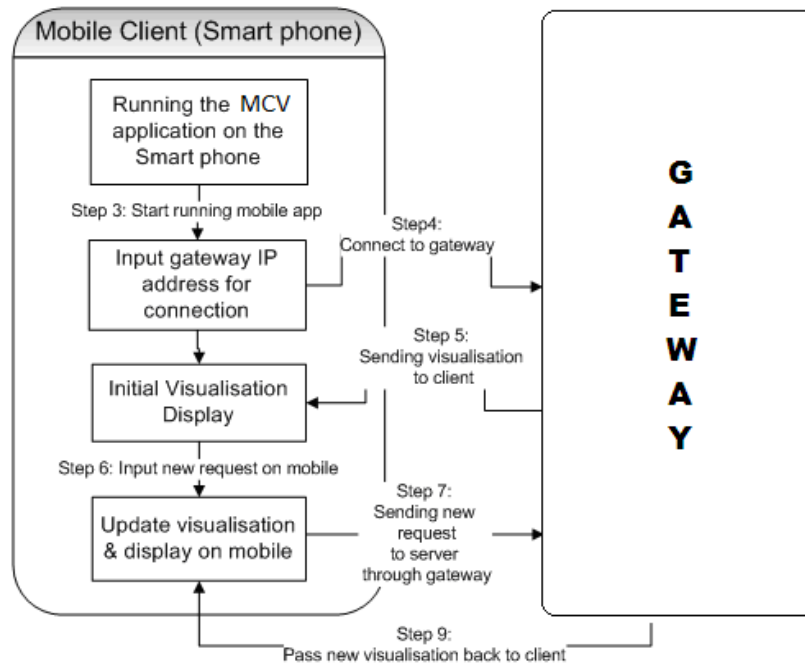


Figure 3-8 Mobile Client flow chart

1. Gateway connection

The mobile client is used to display the visualisation without local rendering. Therefore, the mobile client must establish a connection with the gateway in order to receive the rendered visualisation from the server.

The end user runs the MCV application on the smart phone in order to receive the visualisation from the server and to manipulate the visualisation. The MCV application also enables the mobile client to connect with the gateway (see step 4).

2. Visualisation display

Once the connection between the gateway and the client is set up, the mobile client will be able to receive the initial visualisation generated on the server. In step 5, the end user can use the MCV application to retrieve the visualisation from the gateway through the established connection.

3. Visualisation update

If the end user would like to alter the view of the visualisation on the smart phone, the mobile client will use the gateway to pass the update commands to the server (step 7).

The mobile client will not render the visualisation locally. Step 9 shows that the mobile client can update visualisations remotely on the server and transmit them back to the client via the gateway.

As the main purpose of the thesis is to allow smart phone users to view and interact with visualisations that are generated from a remote visualisation system, the MCV application on the mobile client will have some basic update functions that demonstrate the concept. Those functions are discussed in section 3.6.5.

3.6.4 Communication between server-gateway & gateway-client

The three major components of the design flow are the server, the gateway and the mobile client. Communications between each individual component will be analysed in detail in this section.

1. Communication between server – gateway

The following Figure represents the logical flow between the server and the gateway:

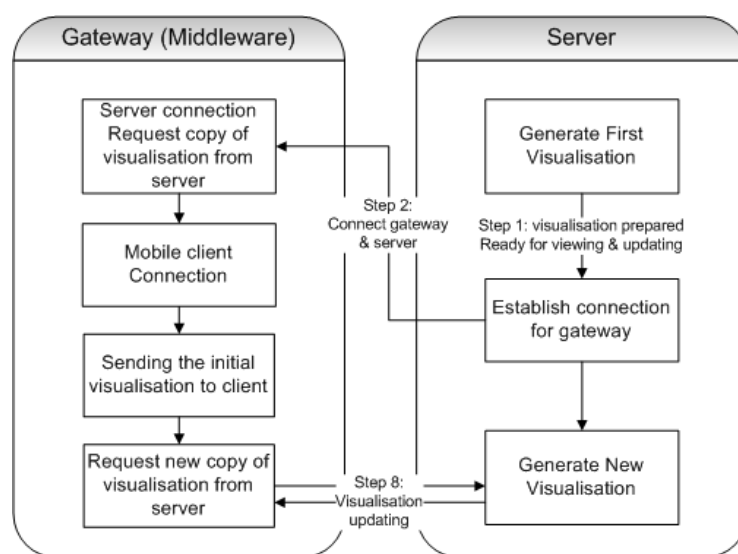


Figure 3-9 Server- Gateway flow chart

Figure 3-9 illustrates communications between the server and the gateway involving steps 2 and 8 in the whole architecture flow.

Once the two sides (the server and the gateway) connect to each other, the gateway will request a copy of the initial visualisation from the server. This copy of the visualisation will then wait to be delivered to the mobile client.

Step 8 in Figure 3-9 represents the visualisation update process between the server and the gateway. It has two-way communication arrows, which means that during this procedure, the two sides will need to exchange information based on the connection established in step 2. This communication is first built from the gateway to the server as the gateway sends update commands (received from the mobile client) to the server through the existing connection. After the server receives those commands, it will create a new visualisation and transmit this visualisation back to the gateway.

2. Communication between client – gateway

The following Figure represents the logic flow between the mobile client and the gateway:

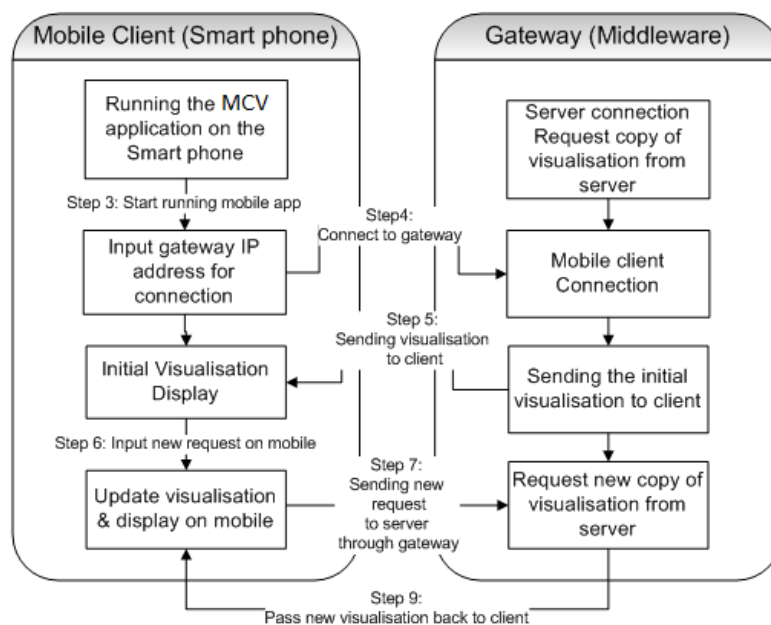


Figure 3-10 Client - Gateway flow chart

There are four levels of communication between the mobile client and the gateway shown in Figure 3-10.

Step 4 communication is the first: the connection between the mobile client and the gateway side. The mobile client makes a connection with the gateway on the smart phone.

Step 5 shows that after the connection is established, the gateway will send the visualisation to the mobile client for display. If the mobile client updates the visualisation, Figure 3-10 shows the communication of new update visualisation flow between steps 7 and 9. The mobile client first sends the update commands to the gateway. After the new visualisation is created by the server, the gateway will transmit the visualisation to the mobile client.

The server – gateway and client – gateway communications include visualisation transmission and updating commands. These interactions will be discussed in more detail in the implementation chapter.

3.6.5 Functions on mobile device

Functionality must be provided to allow the user to manipulate the visualisation via the MCV. To explore the basic concept of viewing and updating the visualisation on the smart phone, it is planned to have the following functions on the smart phone application:

1. Connection

The “Connection” function is used to set up the network connection between the mobile client and the gateway. For example, the end user could input the IP address of the gateway in order to receive the generated visualisation from the server.

2. Rotation, Zoom, Representation

There will be many different functions in the existing visualisation system. In order to have a visualisation application that allows updating, some basic functions of the visualisation system

will be chosen to appear on the mobile application, such as rotation, zooming and changing representation.

When the end user has the visualisation displayed on the smart phone, the rotation function can help the user to rotate the view in different directions. The zoom function will allow the user to alter the size of the viewing object. The representation function is used to switch the viewing object between different representations, such as points or wireframe.

3. Initial View

This function can help users reset the visualisation to the initial view after they make changes to the visualisation on the smart phone. It may be useful if the user wishes to undo some modifications on the visualisation.

3.7 User perspective of application flow

The remote rendering client – server gateway approach should be easy to use. Based on the above description of the system architecture, users will experience the following flow to use a smart phone to view and update the visualisation through the gateway:

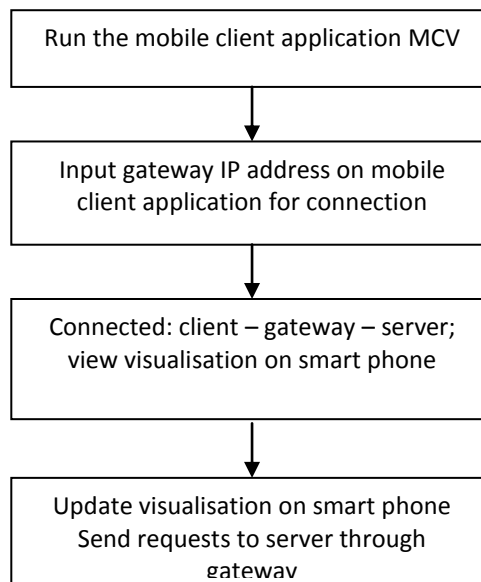


Figure 3-11 User perspective of application flow

From the end user's perspective, the user will only have the smart phone in hand. The server and gateway application are already running and the gateway is waiting for the mobile client to connect. The user will firstly run the MCV application on the smart phone. After inputting the appropriate IP address for the gateway, the network connection is established from the end user side. The user will be able to view the visualisation after this connection is established. By using the functions on the smart phone, the user can send commands to the server through the gateway in order to update the visualisation.

3.8 Summary

In order to meet the requirements for the mobile client application listed in section 3.1, the client – server gateway application is developed for demonstrating the use of a smart phone to visualise a data set. The mobile application allows user to interact with the visualisation remotely within the same visualisation system. The client - server architecture will be able to solve some of the problems of the smart phone, such as limited memory space and lower graphics performance.

The customised mobile application is developed based on the visualisation system by using a gateway approach. The mobile application uses the gateway as middleware to interact with the server and the mobile client. The gateway handles all communications between the two sides. The mobile application does not need to install the visualisation system client on the smart phone and allows a simple end user interface. It also allows the user to control what has been sent from the server to the client and vice versa. Moreover, the functions on the mobile application will not only display the generated visualisation on the smart phone, but also allow users to manipulate the displayed visualisation.

Chapter 4

Implementation

The implementation of the design described in chapter 3 is discussed in this chapter. Based on the gateway approach, this chapter covers detailed system implementation procedures and components for the mobile client application. It also describes how the application works under the smart phone environment, and the technical choices made to implement the design as well as the reasons for those decisions.

4.1 System overview

The mobile visualisation system is described in the design section in chapter 3. It includes three major components: Server, Gateway and Mobile Client. (See Figure 3-4). The server is used to upload data, generate the image and transmit it to the client through the gateway. The gateway is used as middleware to transmit the visualisation result to the mobile client and handle update requests made from the client to the server. The mobile client displays the visualisation and requests new visualisations.

4.2 System environment

A key aspect of the thesis is to show the possibility and test the efficiency of generating and updating visualisations remotely based on an existing visualisation system. In this thesis, we use Paraview (Squillacote, 2007) to create the visualisations.

Paraview is an open-source, multi-platform data analysis and visualisation application. It has its own application programming interface (API) wrapped using Python, that allows developers to interact with the Paraview application. Paraview can be fully scripted using the Python language, which can help developers extend an application (Squillacote, 2007) .

The Paraview system also has its own built-in client – server mode, which allows end users to receive a visualisation remotely on a different PC under this mode. In order to demonstrate the concept of using a smart phone to view and update visualisations, we use Paraview version 3.6.2 client-server mode to generate images for mobile clients.

The server under the Paraview visualisation system is called *pvserver*. In this thesis, the server runs under a Windows operating system. The gateway also runs under the same operating system as the server. The smart phone is an HTC brand Touch Pro2 phone. In this thesis, the operating system on the smart phone is Windows Mobile version 6.5.

4.3 Programming language

To interact with the Paraview system, and to communicate between the server and the mobile client, Python is used to program the gateway for the mobile visualisation system.

The smart phone is the mobile client for end users. Since our smart phone is running the Windows Phone operating system, we use VB.NET to program the mobile client application on the smart phone.

4.4 Gateway implementation

With regard to the design of the gateway, the data exchange workflow through the gateway is illustrated in Figure 4-1.

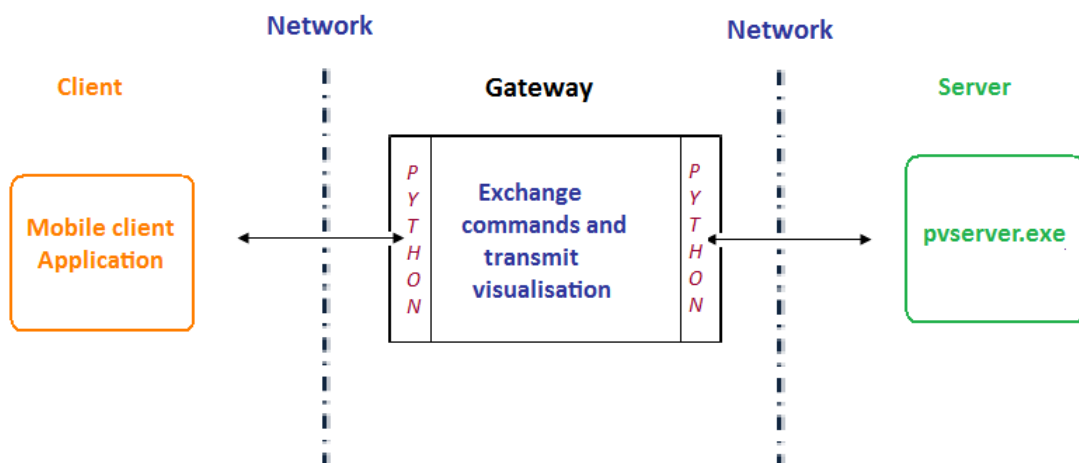


Figure 4-1 Gateway data workflow

It shows the server running an executable program called *pvserver.exe* under the Paraview system. *pvserver* acts as a server to interact with the client through the gateway. The mobile client (smart phone) runs a client application locally. The client application has functions that allow users to connect with the gateway. The gateway runs two python scripts separately. One is for the mobile

client, another is for the server. These two python scripts are used to establish connections and exchange data between the server and the mobile client. Implementations of server-gateway and gateway-client are discussed in detail in the following sections.

4.4.1 Sever-Gateway implementation

We use pvserver to start up the client/server mode between the server and gateway. In this case, the gateway represents the 'client' in the Paraview visualisation system.

A python script called *servergateway.py*, running on the gateway, is used to establish a connection between the server and the gateway.

Now we generate a visualisation using Paraview before transmitting it to the mobile client. To make it happen, when *pvserver.exe* runs, the gateway connects with *pvserver* by running *servergateway.py*. When the connection is made, the gateway accesses the Paraview system, through *servergateway.py*, to allow the server to render a visualisation. Once the visualisation is ready, *servergateway.py* saves the image on the server and waits for a new command sent by the client:

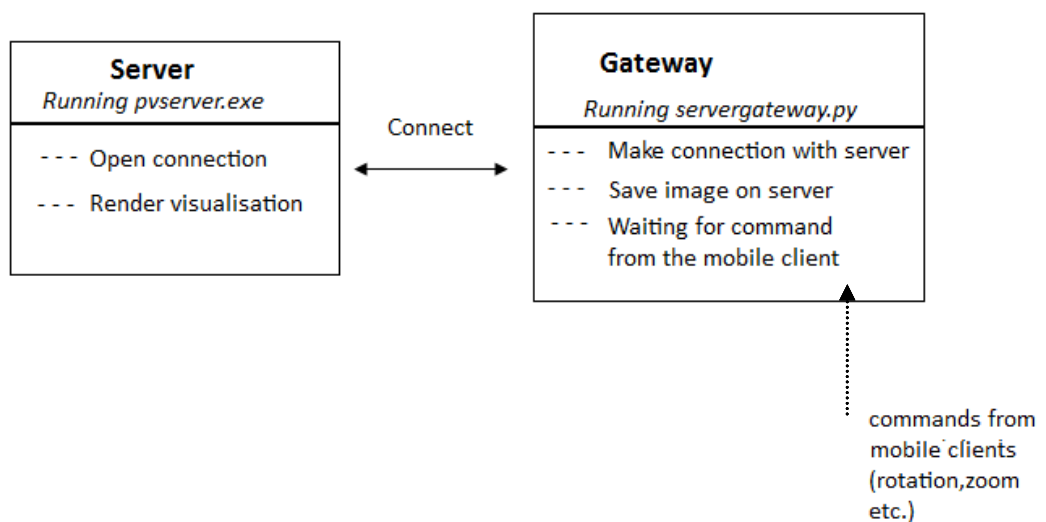


Figure 4-2 Server - Gateway communication

If there are new commands sent from the mobile client, such as rotation or zoom, these commands will be formatted as a new text file each time named "*cmd.txt*".

The following Figure gives an example of the format in the *cmd.txt* file:

```
Commands format:  
30 (vertical rotate)  
50 (horizontal rotate)  
1.5 (zoom)  
Points  
####Reset to first view#### (if none, means no need to reset)
```

Figure 4-3 Commands text file format

The commands in the text file are stored in order so that the gateway interacts correctly with both client and server sides. Following this order, the gateway reads each line and passes the commands to *pvserver* to communicate with the Paraview's *servermanager* module. The server will then render the new visualisation based on these commands and the new image will be delivered to the mobile client through the gateway.

servergateway.py keeps reading the new commands in *cmd.txt* file, once it finishes reading them, it deletes the *cmd.txt* file that contains the old commands it has read already. When new commands are read in the text file, the gateway will pass those commands to *pvserver* to execute and request an updated visualisation.

4.4.2 Gateway-Client implementation

The gateway uses a python script *clientgateway.py* to process interactions between itself and the mobile client. It is used to establish a connection between the gateway and the mobile client, transmit commands and visualisations. The latter two are discussed below.

1. Transmitting update commands

When the end user submits update commands from the smart phone, those commands are sent to the gateway in an appropriate format (see Figure 4-3). The commands must be sent in the correct order because the gateway uses the order to ensure that the server renders the visualisation correctly.

The update commands are formatted in the following order when sending to the server:

- Vertical rotation angle
- Horizontal rotation angle
- Zoom scale value
- Representation selection value
- Reset to initial view or not

One example of the command format is shown in Figure 4-3 under section 4.4.1. The rotation orientation is described in more detail in Figure 4-8 of section 4.5.2.

Once the gateway receives the commands from the mobile client, the gateway uses a file writing function in python to create a new text file named “cmd.txt” (see section 4.4.1). This new cmd.txt file stores commands sent from the mobile client, such as zoom or rotation commands. The commands in cmd.txt file are read line by line. The Paraview server uses those commands to render a new visualisation.

2. Transmitting updated visualisations

The gateway saves every rendered visualisation on the server as a *.png* file. We use the gateway to read the image file and transmit the visualisation to the mobile client through the network connection. The initial visualisation appears automatically on the smart phone once the connection is established by the user.

It is important to save the image files using different names, otherwise, the gateway may send out an old image with the same name. In order to implement this feature, clientgateway.py uses a *name_n_timestamp* function in python to select the latest image file for the gateway to read and transmit. *name_n_timestamp* function gives a timestamp for each image that is rendered and stored on the server. It filters out the latest one from all of the generated visualisations.

4.5 Mobile Client Application Implementation

The mobile client application is designed to interact with a remote Paraview visualisation system for displaying and updating the rendered visualisation. The implementation of this application is discussed below.

4.5.1 Mobile client user interface

The mobile client application was implemented in VB.NET using Visual Studio. The application was tested on a Windows Mobile Emulator 6.0 firstly before running on an HTC Touch Pro2 smart phone. The emulator allows the design or content of the mobile application to be amended relatively easily.

For the purposes of viewing and updating the visualisation on the smart phone, some functions from the Paraview application were chosen to include in the mobile client application. Screenshots of the mobile client user interface are shown below running on the Emulator and HTC phone.

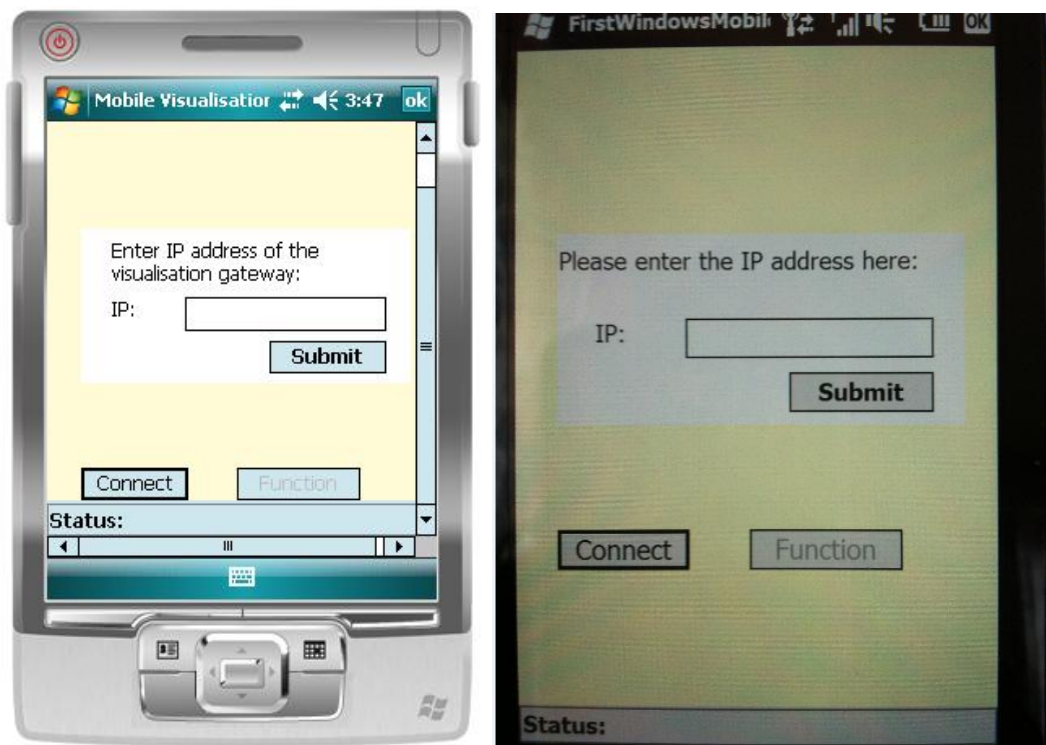


Figure 4-4 Connection Panel of Mobile client interface

The connection panel in Figure 4-4 includes a text field for the user to input the IP address of the gateway. Once the visualisation is generated by the server, the gateway will need to run *clientgateway.py*, then wait for the mobile client to establish a connection with it. On the mobile

client application, users need to input the correct gateway IP address to set up the network connection.

If an invalid IP address is input into the text field, the mobile client application will pop out a message box to alert users:

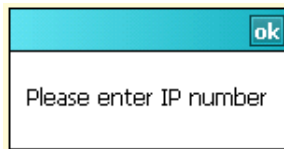


Figure 4-5 (a) No IP address input

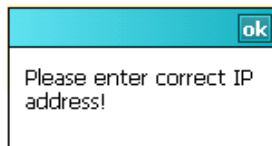


Figure 4-5 (b) Invalid gateway IP address

Figure 4-5 Invalid input alert for gateway connection

These message boxes help users to identify the problems if they cannot connect to the gateway from the mobile client.

Once the connection between the mobile client and the gateway is established, users are able to manipulate the displayed visualisation by choosing from the functions as displayed in Figure 4-6.

There are three command buttons shown in the function panel of the mobile client: *Original View*, *Submit* and *Cancel*. The Submit button is used to send update requests to the gateway after users have finished selecting them. If users have supplied incorrect input, the interface allows users to quit a selection by clicking on the Cancel button. Once the Cancel button is clicked, users are taken back to the visualisation screen and no commands are sent to the gateway.

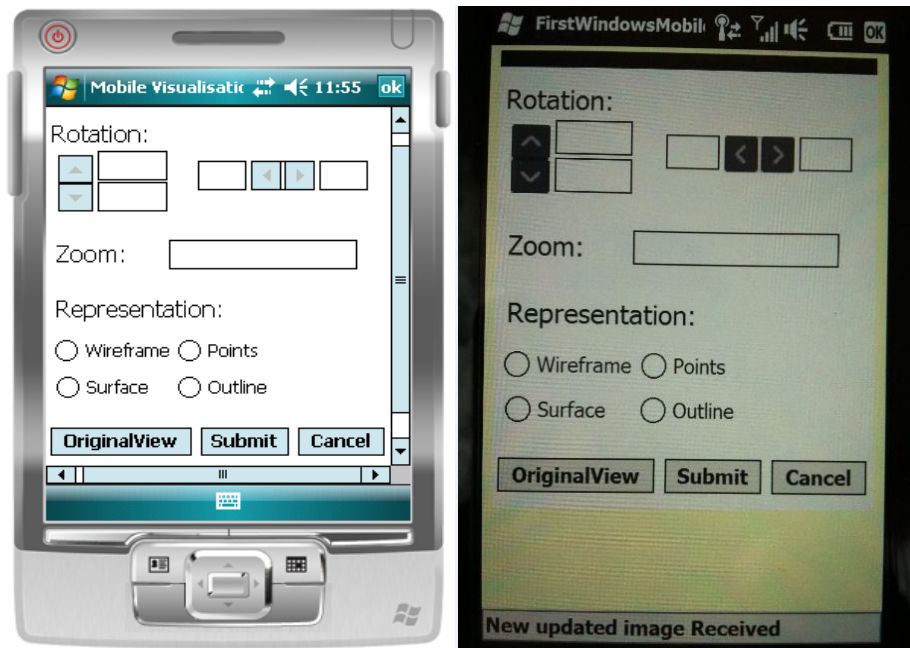


Figure 4-6 Function Panel of Mobile client interface

The Original View button is designed to reset the current visualisation to the initial view. The reason for having this button is because it allows users to undo any of the modifications of the visualisation. It may be convenient to use if users wish to view the first visualisation they worked on.

Other functions in Figure 4-6 are all designed based on the Paraview application. Those functions are described in more detail in the following sections.

4.5.2 Update functions

The update functions include rotation, zoom and representation change.

Before describing these update functions, it is important to know that the Paraview application has a module called *servermanager*, which can be used externally with other platforms. *servermanager* is designed to easily build distributed client-server applications. In this thesis, all the update functions are developed based on this module. In order to manipulate the Paraview visualisation, any interaction between the server and the gateway needs to communicate with this *servermanager* module. In *servergateway.py* script, this module is imported as follows:

```
from paraview import servermanager
```

To establish a connection between the gateway and the Paraview server, the gateway needs to run the following command in *servergateway.py*:

```
connection=servermanager.Connect("localhost")
```

On the other hand, for the update functions mentioned above, we need to be aware that depending on the bandwidth network connection, there will be time gaps when transmitting visualisations from the Paraview server to the mobile client.

➤ Zoom function

Users can apply the zoom function to the current visualisation on the mobile client to zoom in or out. The zoom function requires users to input a zoom value. For example, if a user wishes to resize an object to be 50% larger, the user must input a zoom value as shown in Figure 4-7:

Zoom:

Figure 4-7 Zoom function input

The zoom function is implemented based on the Paraview python class through the *servermanager* module. *servermanager* allows the mobile client to interact with the Paraview server remotely through the gateway python script *clientgateway.py*. While the actual zoom is executed on the Paraview server side by running *servergateway.py*, as mentioned earlier, any visualisation change will need to communicate with the *servermanager* module.

Therefore, we use one of the *servermanager* module's function called `camera.Zoom(ndata)` to zoom the visualisation in the Paraview server. "`ndata`" represents the zoom value entered on the smart phone.

➤ **Rotation function**

Paraview allows users to view the object in a 2D or 3D environment. For the purpose of rotating the object to allow different views, the rotation function is essential to implement in the mobile client application.

To rotate the object displayed on the mobile client, the mobile client needs to send rotation requests first to the Paraview server through the gateway. Once the rotation requests are read and processed by the Paraview server, the gateway will then notify the mobile client and send the image through the network.

The rotation function is designed to have four text fields that allow users to move the camera vertically or horizontally, see Figure 4-8.

The four directions represent camera rotation directions. The mobile client users need to input the number of degrees to rotate the object. The rotation directions are divided into two groups:

a) Vertically

The text fields in Figure 4-8 (a) that are labelled “Rotate Up” and “Rotate Down” mean that the camera position will move vertically for the rotation.

Similar to the zoom function in the *servermanager* module, Paraview has a rotation function called `camera.Elevation(f11)`. “(f11)” represents the vertical rotation value that is input by the user on the smart phone and recorded subsequently in the gateway’s *cmd.txt* file.

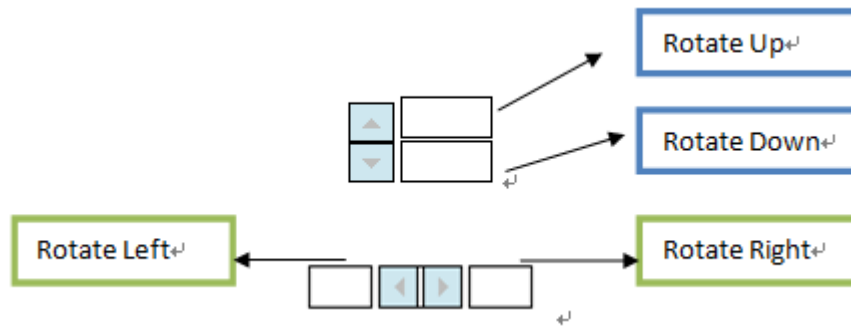


Figure 4-8 (a) Rotation function

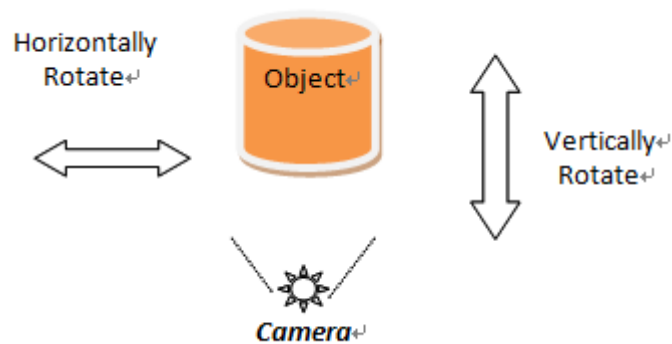


Figure 4-8 (b) Rotation direction for 3D object

Figure 4-8 Rotation diagram

b) Horizontally

In Figure 4-8 (a), the text fields labelled “Rotate Left” and “Rotate Right” mean that the camera position will move horizontally for the rotation. In relation to the *servermanager* module, Paraview has the horizontal rotation function called `camera.Azimuth(f12)`. “(f12)” represents the horizontal rotation value that user is input by the user on the smart phone and recorded on the gateway’s *cmd.txt* file.

Figure 4-9 shows an example of a teapot rotation and a teapot zoom implemented from the mobile client application:

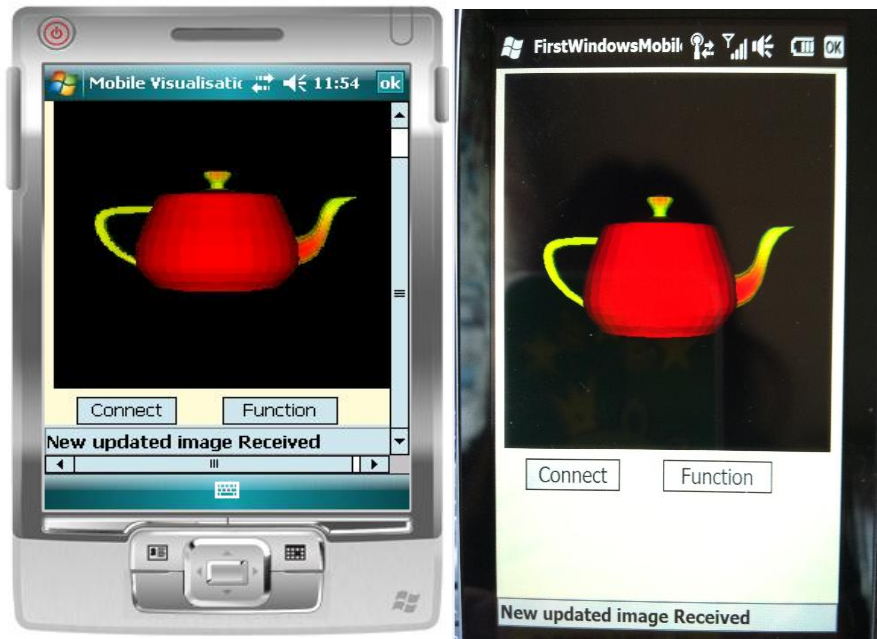


Figure 4-9 (a) Initial view



Figure 4-9 (b) Input parameters

Figure 4-9 (c) Updated visualisations

Figure 4-9 Rotation & Zoom visualisations

Figure 4-9 shows how the original teapot (Figure 4-9 (a)) can rotate and update to the larger one in the mobile client application. Figure 4-9 (b) shows that the mobile client is about to zoom in 1.5 times, move the camera up 45 degrees and then shift the camera to the right by 30 degrees. Figure 4-9 (c) displays the updated visualisation that is transmitted by the server.

In the mobile client rotation implementation process, and as mentioned in the zoom function, the rotation function is implemented based on the *servermanager* module. The rotation degree parameters that are input by the user are delivered in order to the gateway. Once the gateway receives the commands, it writes to a new file named “*cmd.txt*” stored on the gateway. As mentioned in Figure 4-3, the example of the format in *cmd.txt* file corresponding to the user interface.

➤ **Representation function**

The representation function uses the same text file implementation described above to allow users to change representation types for the objects being displayed. In the *servermanager* module, Paraview function `SetDisplayProperties(vtkreader, Representation=rep)` allows users to change the visualisation representation. “*rep*” is the representation value that a user chooses from the smart phone. Users only select one type of representation at a time. The following diagram shows an example of the “Wireframe” representation of the teapot object:

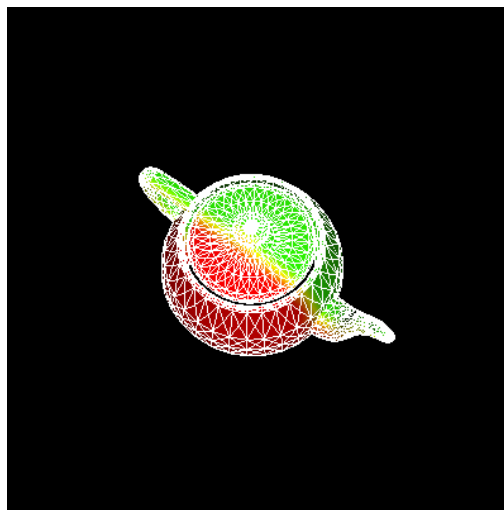


Figure 4- 10 “Wireframe” representation

4.6 Network connection implementation

The communication between server, gateway and mobile client is through network connections.

The gateway is implemented using the same Windows environment as the one that runs the Paraview server. In this thesis, the gateway and the server are running on the same computer. Therefore, the network connection that we need to build is between the gateway and the mobile client.

Transmission Control Protocol (TCP) is a reliable network protocol that is considered to be a safer protocol for file transfer compared with User Datagram Protocol (UDP) as TCP has three steps to confirm that the transferred streams are delivered from server to client, while the UDP delivers streams but do not require feedback from the client side. In this thesis, the gateway needs to handle text and image files between the server and the mobile client. Although the TCP protocol requires acknowledgement from the receiver side which may take longer time than UDP, to ensure the streams are delivered properly and safely, it is still a good choice to use for implementation with network interaction.

To enable communication between the server and the client using TCP, the server application needs to create a listening socket on a TCP port and wait for the client to connect. Once the client connects to the port, the server accepts the connection and starts to respond to the client.

From the TCP client/server perspective, to build a connection between the gateway and the mobile client, the gateway acts as a “server” and the mobile client acts as a “client”.

In this thesis, the mobile client needs to enter the IP address of the gateway on the mobile interface. First, the gateway needs to create a new listening socket on a TCP port and wait for the mobile client to reply. The mobile client can then enter the gateway IP address on the smart phone. Once users click the submit button under the connection panel, the connection should be established.

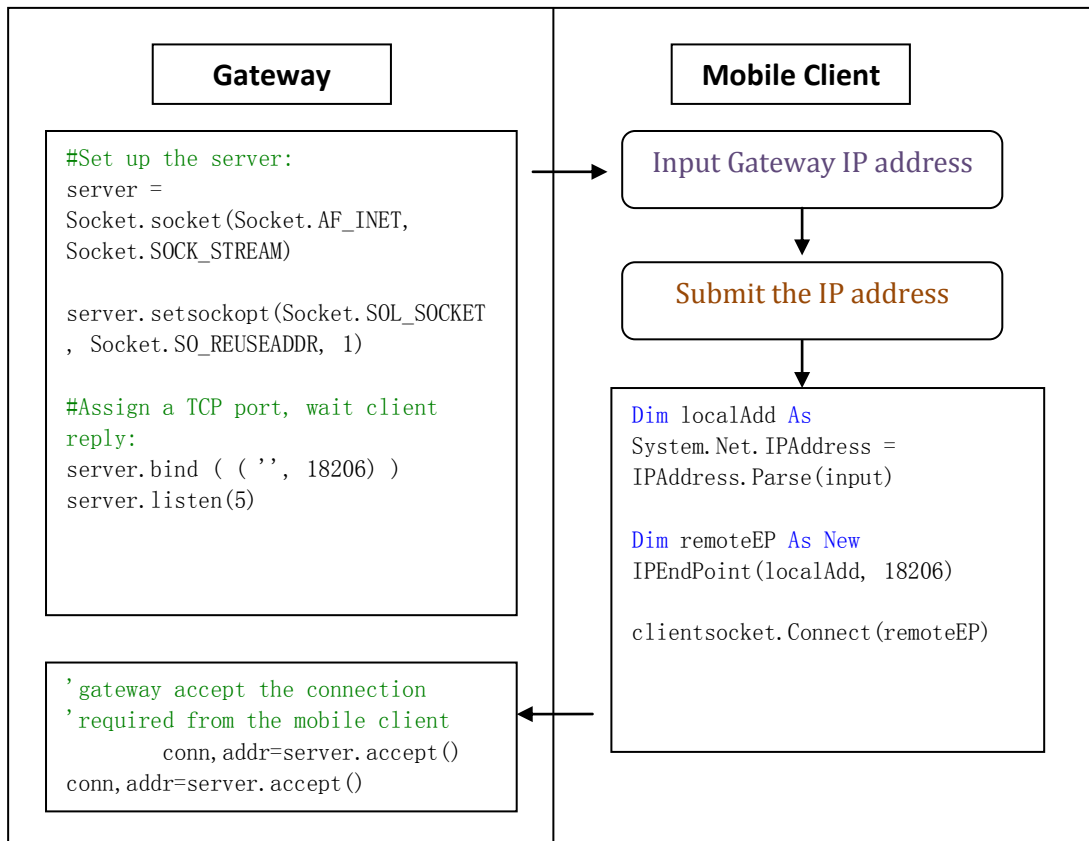


Figure 4-11 Gateway – Mobile client network connection

Figure 4-11 includes part of the python script *clientgateway.py* on the gateway side. The client side uses VB.Net code for connecting with the gateway “server”.

The gateway part in Figure 4-11 runs a python script called *clientgateway.py*. This uses a TCP socket to set up a connection between the gateway and client.

In Figure 4-11, the TCP port number (“18206”) on the server side needs to bind with the same number on the mobile client application before users submit the IP address, otherwise the mobile client will not be able to successfully connect with the gateway “server”.

In addition to the initial network connection between the gateway and the mobile client, the mobile visualisation system should allow users to connect with the gateway at any stage as long as the

server from the Paraview system is available. *clientgateway.py* has a class called *ClientThread*. Figure 4-12 shows part of the python script in the *ClientThread* class:

```
'allows the gateway keep the TCP connection open
'even after the mobile client disconnected with the
gateway
class ClientThread ( threading.Thread ):
    def __init__ ( self, channel, details ):

        threading.Thread.__init__ ( self )
        self.channel = channel
        self.details = details
        self.thread_stop = False

    def run ( self ):
        global connection
        print 'Received connection:',
        .....
        .....
```

Figure 4-12 Client infinite loop in python

This class works as an infinite loop that keeps the gateway TCP port open and waits for users to re-connect with it. In Figure 4-12, “*self.thread_stop = False*” means that the gateway itself will not stop running and keeps giving the signals for the mobile client to connect with the gateway.

4.7 Summary

Based on the Paraview visualisation application, this chapter presents the techniques that have been used to implement the server, the gateway and the mobile client applications. The server and the gateway run on the same desktop computer environment under a Windows operating system. The mobile client application was implemented in VB.NET. The application was tested on a Windows Mobile Emulator before running with an HTC Touch Pro2 smart phone under Windows Mobile version 6.5.

The server in this thesis uses the built-in *pvservers* in the Paraview client/server mode. Two python scripts *clientgateway.py* and *servergateway.py* were implemented to make connections and exchange commands between the mobile client, the gateway and the server. The thesis used a TCP

socket connection to ensure commands and visualisations are transmitted correctly through the network.

A visualisation may be modified using the mobile's user interface to rotate, zoom, and change the representation of the display. The mobile client is able to update the visualisation by interacting with the server through the gateway.

Chapter 5

Evaluation

This chapter discusses the evaluation of the system, including the usability of the mobile client application and examines the functions of the application. It also discusses its successes and limitations.

First, the evaluation environment is introduced, followed by the evaluation procedure. We include a summary of the findings and a discussion of the results.

5.1 End user trial

The solution developed used a gateway approach to act as middleware between the mobile client application and the server. This reduced the workload on the mobile client. This solution also aimed to make the mobile application interface as simple as possible to generate and update visualisations on request from the smart phone. The teapot object in Chapter 4 was selected from the Paraview application data for this end user trial, which was designed to evaluate the gateway approach.

The trial involved several tasks for users to complete. Before the end user trial, a pilot test was carried out in order to get feedback on the design of the trial before the formal evaluation. The aim of the end user trial was to demonstrate the feasibility of the existing Paraview visualisation software to communicate with a mobile device. The main evaluation will be to evaluate whether a user can efficiently use a smart phone to update visualisations generated from the Paraview system, compared with using a desktop computer.

We conducted the evaluation based on the following aspects:

1. Whether or not the image generated through the Paraview server was successfully sent to the smart phone through the gateway.

2. Whether or not the visualisation was displayed correctly on the smart phone.
3. How fast the image was sent through the network from server to mobile.
4. Whether the client received the expected updated results.

5.1.1 User trial environment

For this end user trial, there were sixteen participants in total. The participants included university postgraduate students and general workers in different industries. All tests were conducted during normal working hours (9:00 – 17:00) and under stable network conditions.

Before installing the mobile application on the HTC smart phone, it was tested first using the Windows Mobile Emulator 6.0. We first ran a pilot test before the actual end user trial to test the usability of the mobile application. Observations throughout the pilot test helped the researcher to identify some problems regarding the design of the evaluation. We used the pilot test to improve the evaluation process in order to obtain more useful information during the trial. After two volunteers took the pilot test, the mobile application was ready to test on the HTC phone via the end user trial. The following Table 5.1 describes the environments for the mobile and desktop trials:

Table 5-1 User trial environment

Machine	Platform	Monitor	Software
Laptop Processor: Intel T7250 RAM: 3GB	Windows Vista Home Premium	15.4" wide screen	Paraview version 3.6.2 gateway python script
HTC Touch Pro2 Smart phone mobile Processor: MSM7200A RAM: 288MB	Windows Mobile 6.5	400x800 Resolution 3.6" touch screen	Tested Mobile application

During the user trial, when testing using the smart phone, the laptop acted as a server with the gateway running on the same laptop environment.

5.1.2 User trial procedure

The end user trial procedure included three major steps:

1. Introduction programme:

This was designed to help trial participants become familiar with the mobile application functions and the Paraview system. It included written instructions to guide participants through the tasks.

2. Mobile and desktop tasks:

After the participants had become familiar with the mobile and desktop applications, they were given two sets of tasks to complete. In order to counter learning effects during the trial session, half of the participants were asked to complete desktop tasks first, the other half completed mobile tasks first.

While they were carrying out the tasks, the participants were observed (with their permission) and the time to complete each task was recorded.

The mobile and desktop tasks (see Appendix A & B) were chosen according to the four aspects described earlier. The tasks were chosen to demonstrate the feasibility of a mobile device interacting with the visualisation system based on the following factors (for task explanations see section 5.2.1):

- Server-client connection
- Initial visualisation is displayed correctly prior to any user input
- Visualisation interaction, including functions such as rotation, zoom, and different representations
- Return to the original visualisation after interaction

3. Questionnaires and interviews

Once the participants had completed both their mobile and desktop tasks, questionnaires were given for them to complete (see Appendix C). At the end of the trial, the participants were interviewed

based on their experiences of using the mobile application and to gather any suggestions for improving the mobile application.

5.2 Results & discussion

The participants were asked to carry out the user trial using both smart phone and laptop. Before the user trial started, they were given instruction to get them familiar with the smart phone and the application platform, such as how to enter the numbers on the touch screen phone and the meanings of the different function buttons on the mobile application.

5.2.1 Observation

Participants were observed while they were completing the tasks. The observations were divided into two sets of results: mobile tasks observations and desktop tasks observations.

Mobile task observations:

- Task 1 : Create a connection with ParaView

In this task, participants were required to input a given IP address onto the smart phone application to establish a connection between the server and the smart phone. Some participants were not familiar with touch screen smart phones, especially when inputting the gateway IP address for the network connection as the smart phone has a much smaller screen compared with a desktop monitor.

- Tasks 2 & 3 : Rotate the visualisation to find the colour of the teapot body and lid

Tasks 2 & 3 asked participants to evaluate the rotation function. These two tasks were used to evaluate whether users were able to rotate an object in different directions, such as horizontal and vertical. Many participants tried to click the up or down button on the mobile application to rotate the object instead of inputting degrees. A significant number of participants spent time on working out the rotation directions and angles.

- Tasks 4 & 5: Make the teapot two times bigger / half size smaller than the current view

Participants were asked to enlarge or reduce the visualisation by inputting the scaling factor under the zoom function. Some participants were confused about the scale factor when conducting task 5

(see more details in section 5.2.3). One participant tried to directly rotate and zoom the object on the screen using fingers rather than input parameters.

➤ Task 6: Put phone into landscape mode

This task is used to test whether the different orientations of the smart phone will affect the visualisation display. During the trial, some participants forgot to use the introduction guide for the use of the mobile application when they tried to complete this task. They looked for a landscape function on the screen rather than simply rotate the device instead.

➤ Tasks 7 & 8: Representation change and reset to initial view

Task 7 required participants to change the render representation of the current visualisation, such as wireframe presentation, surface representation etc. Task 8 asked participants to return to the original visualisation whenever it was necessary. None of the participants had any problems when completing these two tasks.

Desktop task observations:

➤ Tasks 1 & 2: Connect with server & load teapot file from Paraview server

Similar to mobile task 1, desktop task 1 required the participants to establish a connection between visualisation server and the desktop computer. Desktop task 2 asked the participants to load a data file and generate the visualisation. Most of the participants easily completed these two tasks. However a few tended to spend time on searching for the data file in an incorrect location when completing task 2 (see Figure 5-2 in section 5.2.2). The instruction of how to search for the file was outlined in the given manual. Some forgot to look at the instruction when completing task 2.

➤ Tasks 3 & 4: Rotate to find the certain colour of the handle and the body of the teapot

These desktop rotation tasks were easily completed by most participants. However some of the participants forgot to look at the instruction guide that was given to them before the trial started. Therefore, those participants spent more time looking for the rotation function on the desktop visualisation application (see Figure 5-2 in section 5.2.2).

- Tasks 5 & 6: Make the object view approximately two times bigger/half of the size

Most of the participants found those zooming tasks confusing as the Paraview desktop application does not have a function that allows users to zoom exact multiples, such as 2 times bigger or smaller than the original object. The participants had to zoom approximately based on their visual look.

- Tasks 7 & 8: Change the representation of the visualisation & return to the initial visualisation

None of the participants had any problem when changing the representation of the visualisation in task 7. However, when some of the participants came to view the initial visualisation in task 8, they tried to find a button on the Paraview application that could automatically reload the initial visualisation. But as the Paraview application does not have this function similar to the mobile application, they had to manually delete and reload the initial view themselves.

5.2.2 Completion Time

For each participant in the trial, the time was recorded for completing each individual task for both mobile and desktop version:

Table 5-2 Average completion time for tasks

Task	Mean Time (seconds)	
	Desktop	Mobile
Connection	29.8	36.5
Rotation	51.1	111.2
	36.0	51.1
Zoom	35.1	30.3
	19.1	36.6
Representation	45.0	30.7
Reset	57.9	19.6

Table 5-2 above shows the mean completion time (seconds) for all participants that in order to accomplish different tasks in the different environments. The following paragraphs illustrate the results in more detail.

Mobile task time

The average completion time for each mobile task is shown in Figure 5-1(a):

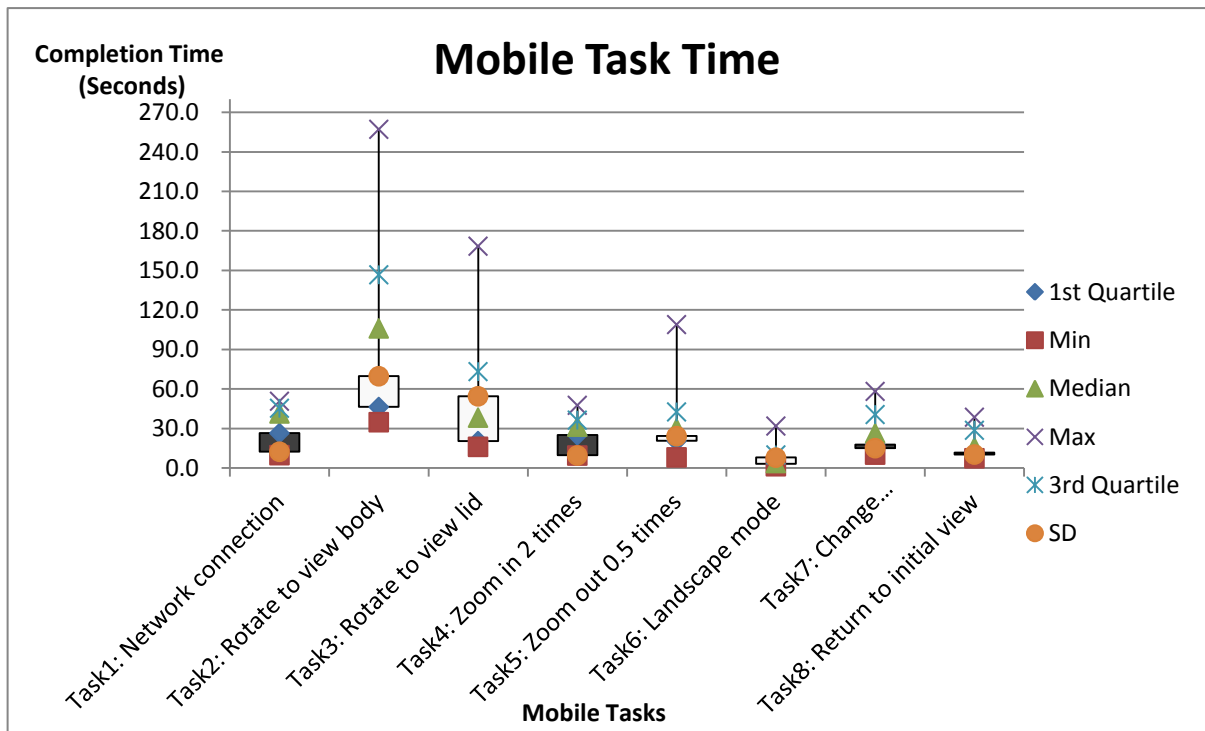


Figure 5-1 (a) Average Completion Time for mobile tasks

Figure 5-1 (a) is a box and whisker plot chart. It shows the inter-quartile range of time the participants' completion times for different mobile tasks. The horizontal axis represents the different mobile tasks that participants completed. The vertical axis represents the completion time for each task (measured in seconds). It clearly shows that participants spend most of the time on completing tasks 2 and 3 (the max time spend on these two tasks), which involved rotating the object to find a specific colour.

The transparent box in Figure 5-1 (a) indicates that the values of the variances of the completion times for the particular tasks. The box shows that participants were spending more time on task 2 and 3 (rotation tasks) than other tasks through the larger box area range. In other words, it means that some users may have issues completing those tasks since the time to finish the two tasks were widely variable between users. This is backed up the previous observation section in relation to the rotation task.

There are two reasons that made task 3 more difficult to complete than others:

1. Some participants did not have any education studying visualisation. It meant that they were not familiar with 3D object rotation. Thus it would take more time for them to understand how the rotation button and input parameters work in the mobile application.
2. For those participants who had a visualisation background, they also needed time to test out how many degrees they had to input in order to generate the visualisation as they expected.

Ease of completing mobile tasks in different order

During the user trial session, half of the participants were asked to complete mobile tasks first before they started to work on the desktop tasks.

Based on the results of the questionnaires, the following diagram illustrates how different orders may affect the ease of mobile task completion:

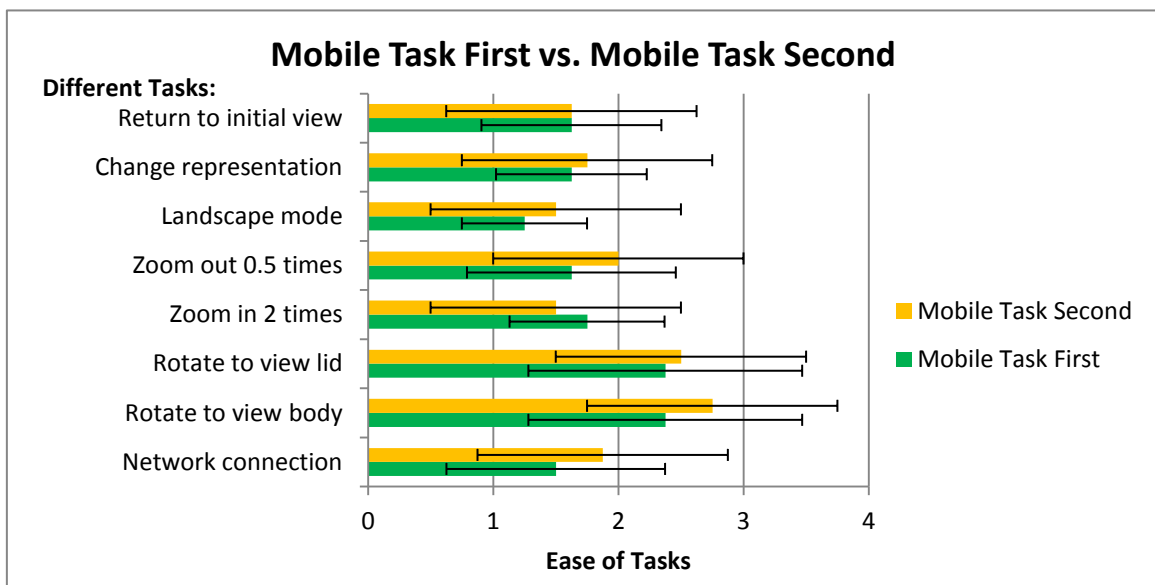


Figure 5-1 (b) Mobile tasks first vs. mobile tasks second

Figure 5-1 Mobile tasks comparison figures

The numbers on the horizontal axis (range up to 3) represent ease of tasks. The rank scale is based on the data collected from question 8 in the given questionnaire (see Appendix C). All the participants were given a 5-point scale and an optional non-completion option to rate the ease of different tasks: “1” represents “Very Easy”; “2” represents “Easy”; “3” represents “Neutral”; “4” represents “Difficult”; “5” represents “Very Difficult”.

Figure 5-1(b) shows that half of the participants who were asked to complete mobile tasks first seemed more likely to rate “Easy” or “Very Easy” compared with the other half participants who were asked to complete desktop tasks first.

Desktop task time

The completion time for each desktop task is illustrated in Figure 5-2:

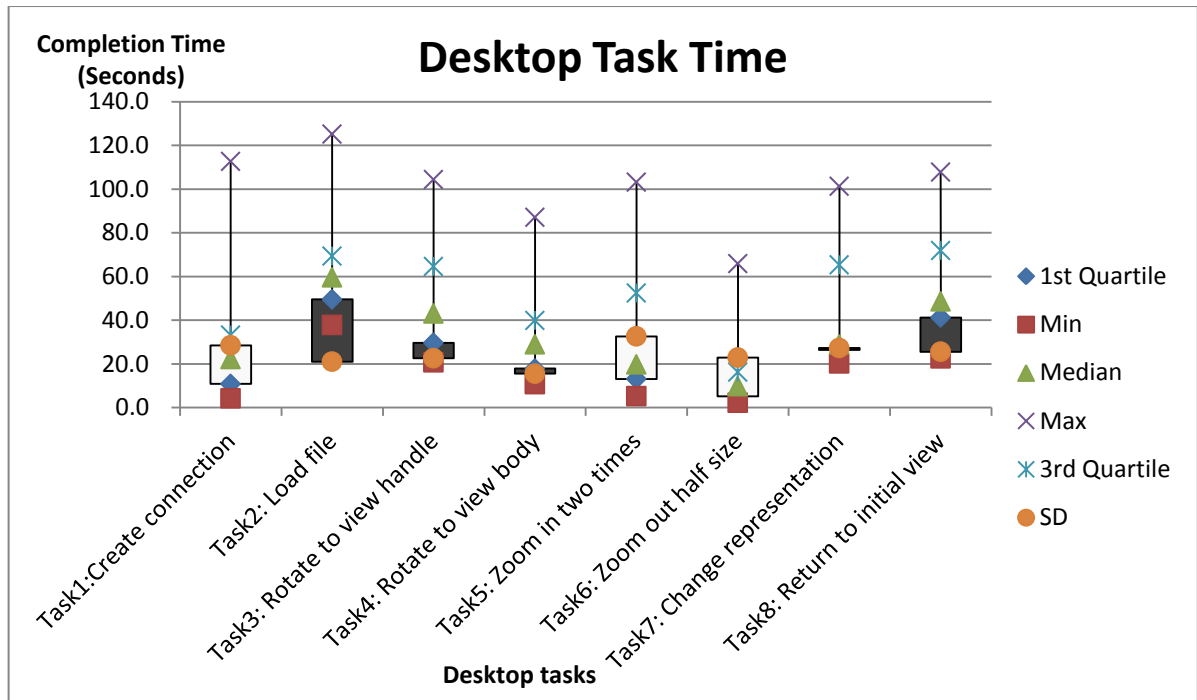


Figure 5-2 Average completion time for different tasks using Desktop

Similar to the mobile task trial, the time was recorded to show which tasks took longer than others in the desktop version. The horizontal axis represents different desktop tasks that participants completed in the trial. The vertical axis represents the completion time for each task (measured in seconds). Figure 5-2 shows that tasks 2 took most time to complete during the desktop user trial. Task 2 required participants to load a file from the Paraview server. Compared with completing other tasks, some participants took longer to locate the specific file. Instead, they tried to look in other directories to locate the file.

The Median value in Figure 5-2 also indicates that in average, task 8 took longer time for participants to accomplish besides task 2. Task 8 needed participants to go back to the initial visualisation. In this

case the participants were required to reload the visualisation as the Paraview application does not have a simple button to reset the current visualisation view to the original view.

Figure 5-2 indicates that some users may have issues completing task 5 (the box for task 5 is the largest box) since some individuals spent longer on the task as they zoomed multiple times. Observations showed this is because that some of the participants were searching for the zoom function on the Paraview desktop application to select or input the zoom value. They thought the desktop application had a same function as the mobile device to allow them input the zoom value.

In addition to the above explanation and observation section, some participants forgot to look up the introduction manual when completing the desktop tasks in the trial. As a result, some participants that do not have any background in visualisation actually completed the tasks faster than the ones who tried to accomplish tasks following their own routines.

5.2.3 Questionnaire results and discussion

After completing the tasks, the participants were given questionnaires. (See Appendix C) These indicated usage of smart phones as follows:

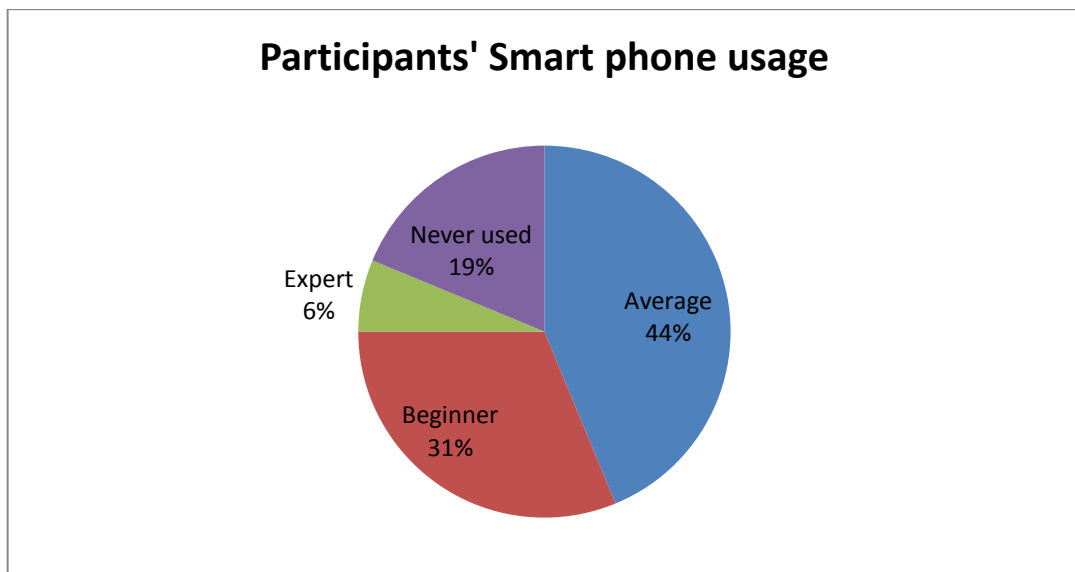


Figure 5-3 Smart phone usage among participants

Figure 5-3 shows the percentage of familiarity with smart phones among all the participants. Most of them have used a smart phone, and most ranged from beginners to average. In particular, only one participant rated as "Expert", shown as 6% in Figure 5-3 (one out of sixteen participants) for using a smart phone. There were a portion of participants (19%) that have never used a smart phone before.

In order to know whether participants were familiar with using a smart phone to receive a file, Figure 5-4 illustrates the frequency of using a smart phone to download a file:

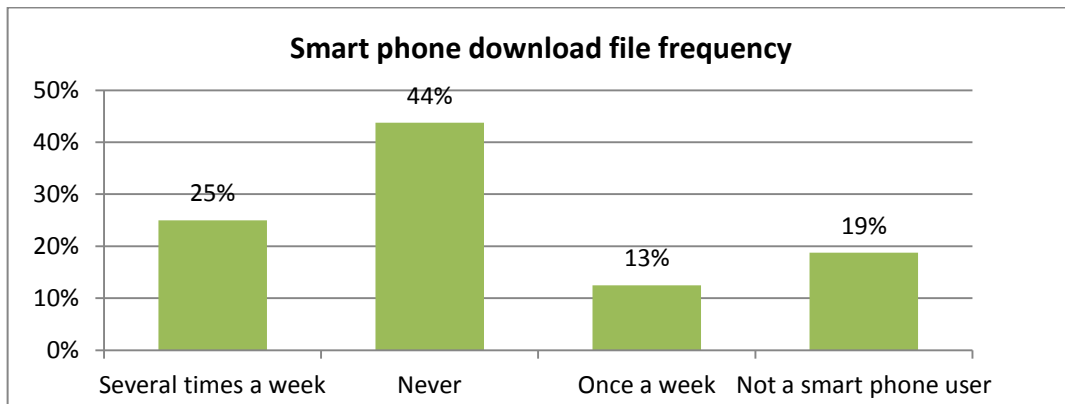


Figure 5-4 Smart phone download file frequency

Figure 5-4 shows that most of the participants (44%) have never used a smart phone to download a file even though they use smart phones. 38% of the participants know how to download a file on a smart phone. Since 19% of the participants have not used a smart phone, therefore, they do not know how to download a file on a smart phone.

For those who participated in the user trial, Figure 5-3 shows that most of the people have used a smart phone before. As the thesis is aimed at using a mobile device to share visualisations, using a mobile visualisation application to directly interact with the visualisation is more in line with users' experiences and expectations than just viewing a visualisation via the download of a static image.

Easiness

All the participants were asked to rate the ease of completing different tasks. Figure 5-5 shows the responses.

Figure 5-5 shows that all the participants finished the mobile tasks. Most of the participants rated "Easy" or "Very easy" for the different mobile tasks. Tasks 2 and 3 (see Figure 5-5 (b), (c)) were rated more difficult compared with task 4 to 8 (see Figure 5-5 (d-h)).

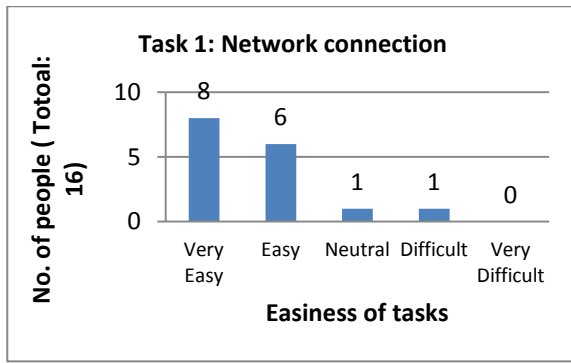


Figure 5-5 (a)

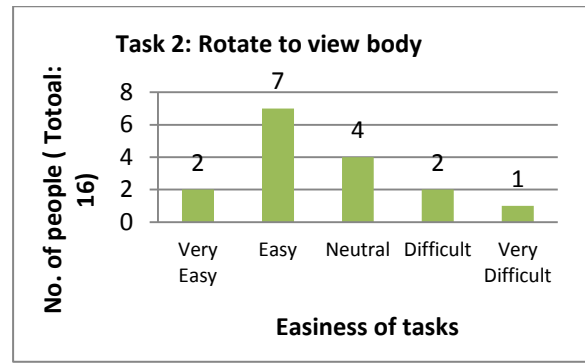


Figure 5-5 (b)

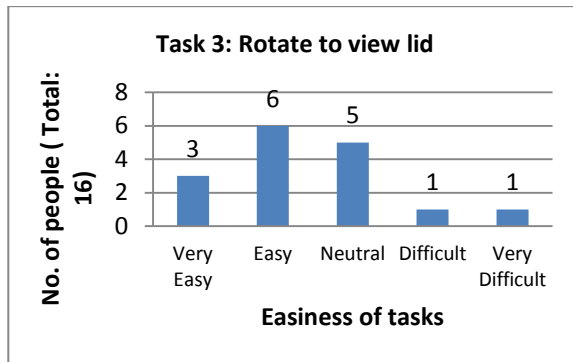


Figure 5-5 (c)

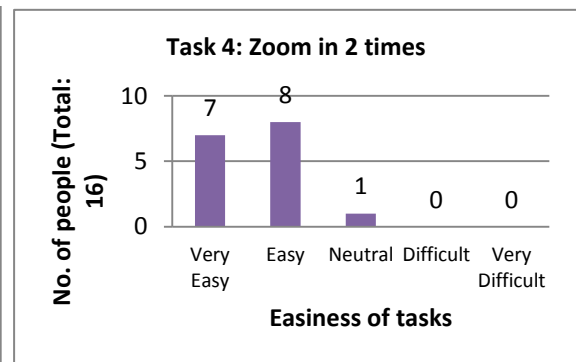


Figure 5-5 (d)

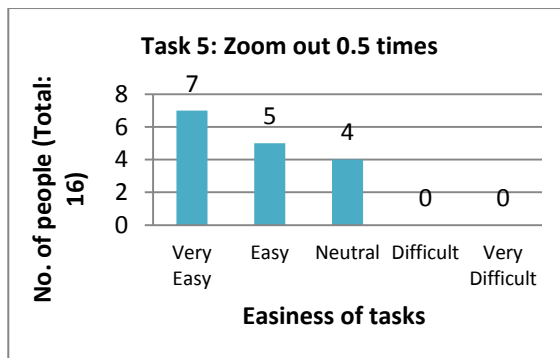


Figure 5-5 (e)

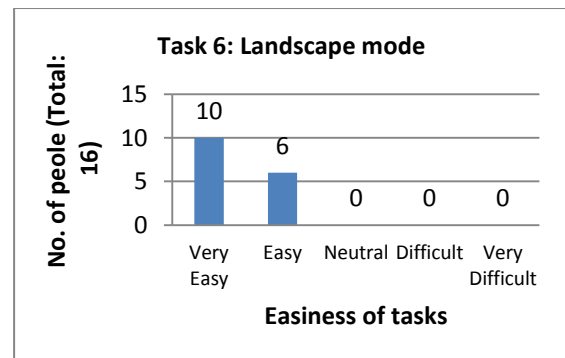


Figure 5-5 (f)

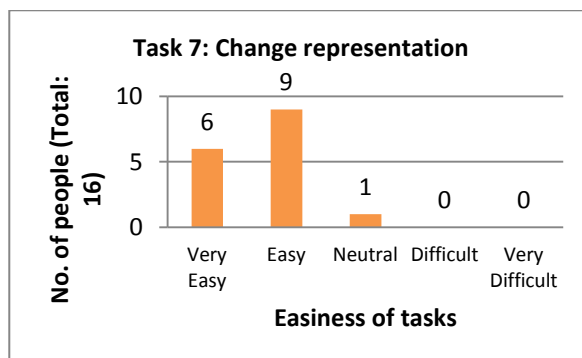


Figure 5-5 (g)

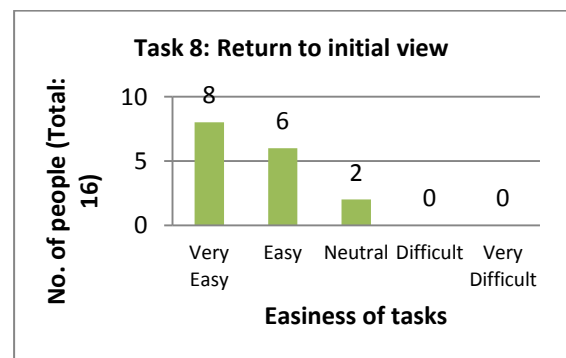


Figure 5-5 (h)

Figure 5-5 Easiness of completing Mobile Tasks

In particular, participant 2 rated “Difficult” for network connection and “Very Difficult” for rotation tasks. This participant was not familiar with using a smart phone, and input of the gateway IP address for a network connection was the first time she had used a touch screen keypad on a small smart phone. As a result, the participant spent more time than average in completing mobile task 1 (network connection) and found it difficult to complete.

When completing the mobile rotation tasks, participant 2 was confused about the direction of rotation and tried to click the “up or down” control instead of inputting the number of degrees to rotate. The participant stated that rotating the object was the most difficult task among all the mobile tasks, commenting:

“Rotating the teapot --- it is mostly about ‘trial & error’ as we have to ‘guess’ the degree of rotation of the object”

Participant 3 also rated “Difficult” for rotating the object. When completing this task, the participant tried to drag two fingers across the display to rotate the object in real-time rather than inputting rotation degrees. Participant 3 was also confused about how many degrees should be input in order to get an expected view and commented on rotation being the most difficult mobile task as follows:

“Rotating, because it is hard to find a suitable value for rotating”

Participant 8 also chose rotation as the most difficult mobile task, commenting:

“Rotating --- to decide the angle & the degree (number)”

Participant 15 input a sequence of small values for degrees, such as 1, 10 and 13 to rotate the teapot. Consequently, it took him a much longer to rotate the teapot compared with other participants. In addition, as the rotation task required a particular part colour of the object to be found, participant 15 spent more time adjusting the view of the object in order to see the other side of body very clearly. The participant commented:

“Finding the second colour on the body of the teapot is most difficult. I did not rotate it far enough initially.”

In relation to the rotation difficulties, there is more discussion in section 5.2.4 and suggestions for improvement in section 5.2.5.

All the participants successfully completed the mobile tasks and wrote down correct answers for the mobile rotation tasks. Figure 5-5 shows that most of the participant rated “Easy” or “Very easy” for tasks 4 to 8, which including zooming, changing representation, viewing the object in landscape mode and returning to the initial view.

The results for task 5 (zoom out 0.5 times) in Figure 5-5 show that 4 participants rated it “Neutral”. The observation notes state that this was because of confusion about “0.5 times” as the 4 participants thought that the object had already been zoomed 2 times bigger, so that 0.5 times would mean a final scale value of “1” instead of “0.5” as required. After those participants entered the scale value of “1”, they found that the visualisation had not changed once they submit the request. Then some started to look over the instruction manual again and some amended the scale value to “0.5” to see whether the visualisation would reduce to half of its size. This misunderstanding is easy to solve if the zoom function is explain more carefully under the instruction session before the trial.

Advantages of the mobile application

There are many advantages of using the mobile application. The answers from the participants can be summarised as the following bullet points:

- Accessibility, mobility, and convenience

The participants pointed out that using the mobile application would allow them to access visualisations in different locations. By using the mobile application, the participants do not need to keep data files locally as all the data is stored on a server. The smart phone is easy to carry around. It is convenient to use the mobile application when away from the office.

- Simple interface

One participant commented that:

“Main functions appeared to be simplified.”

Some participants mentioned that the mobile application is simple to use as all the functions are on one screen and are easy to control. The mobile application is straight forward and has clear buttons. In particular, the mobile application has a function called “original view” to allow the participants to return to the initial view without any complex steps, such as deleting the view and reloading it as in the desktop application.

➤ Mobile compatibility

The mobile application is developed based on a desktop application. One of the participants thought it was good to have the same software that can be used both on the mobile device and the desktop.

➤ Data validation check

When some participants input invalid values into the mobile interface, the application automatically pops up a message window to alert them. Through the trial observation, it was found to be very useful and helped the participants correct the invalid data.

Disadvantages of the mobile application

➤ Small screen

One of the major disadvantages of the mobile application seemed to be a physical issue: the smart phone’s small screen. This was noted especially when changing the orientation: the landscape screen needed to be scrolled down to view all the control buttons. Also one participant claimed that:

“Images can be too small for advanced editing tasks”

➤ Rotation function

Most of the participants found that the rotation function was difficult to use. They normally failed to input appropriate degrees to generate the image at the first attempt. They needed to try a couple of times to find the correct input value for rotation.

➤ Slow

Some participants stated that using the mobile application to complete some tasks on the smart phone was slower than using a desktop computer. Since the desktop tasks were carried out based on the desktop computer which had the server installed within the computer, the smart phone was in a remote location to access the server. Meanwhile, the desktop computer is faster running applications due to its advanced computer processor (see Table 5-1). It is fair to say that the slower processing time for some tasks would be expected when using the smart phone.

➤ Fewer functions compared with desktop

One participant claimed that the desktop application had more functions compared with the mobile application. This is because the purpose of developing this mobile application is to test out the concept of updating visualisations remotely within an existing visualisation system by using a smart phone, rather than providing a full range of functions.

5.2.4 Interview results and discussion

After the participants had completed both desktop and mobile tasks, they were interviewed based on their experiences of using the mobile application compared with the desktop application. The interview focussed on investigating the ease of use of the mobile application; any preferences between using the mobile and desktop applications; and suggestions for improving the mobile application.

Ease of use of mobile application

Many participants thought it was quite easy to use the mobile application. However there were five participants who had different answers regarding this question. Of these, three had a similar answer, which was:

“It would be much easier if the number of degrees to input was known when completing the rotation task.”

One participant preferred to use a bigger screen and a mouse and therefore found it difficult to use the mobile application:

"It is complicated. If only view the image on smart phone, it is fine. But I prefer to use bigger screen and mouse to alter the object"

One participant found the rotation process difficult to use:

"It is quite difficult. The rotation part is hard, not sure how much degrees to input, prefer mouse click as in desktop application."

It seems that this participant tried many times to input small numbers, such as one, ten to rotate the object. This makes it even harder for the participant to input a suitable value for "degrees" in order to view the object, as the rotation changes are too small to see on the small screen of the mobile device.

Preference: mobile or desktop application

When asked about the participants' preferences when comparing the mobile and desktop applications, four out of sixteen participants chose the desktop application because of the bigger screen and use of keyboard for input. The following are comments on choosing the desktop application:

"Easy to manipulate on big screen and use mouse"

"Prefer mouse and keyboard, more detail on desktop application"

"Always use desktop screen, mouse and keyboard"

Five participants stated the mobile application as their first choice especially outside of the office even they can use the desktop application. Some of the reasons are similar, which can be described as following:

"Simple, much easy to access data"

"Straightforward, easy to use"

"Mobile is easy to carry around and no need to store lots of data"

The rest of the participants stood the line between the mobile and desktop application. Some of them would choose depending on the situation and tasks:

"The choice depends on where I am and which task"

“Much easier to view image on mobile, but if I need to alter detailed object, will choose computer”

The other participants pointed out that they would prefer the mobile application if the rotation functions could be improved.

Suggestions on improving mobile application

After the participants completed the questionnaire, a short interview was conducted to collect feedback based on their experiences. Suggestions were collected based on the participants' interview notes and questionnaire results.

Most of the participants stated that the rotation task was not easy to complete because they needed to input suitable values to view the object. They made some suggestions on how to improve the rotation function on the mobile application.

From the participants' experiences, there are two major aspects to consider:

➤ Input value

The participants were not sure of the degree value that should be input in order to generate a required image. Especially for more complex objects other than the teapot, such as mechanical parts, it is difficult to guess the appropriate input value for rotation.

➤ Update a new rotated image

For each rotation task, the participants needed to update the image by clicking more than one button and could not view the new image immediately.

Based on the above, the participants suggested that it would be better to rotate the object on screen by using a finger or pen similar to the desktop application which uses a mouse to rotate in real time.

The main idea was to change the current way of inputting the value for the rotation angle. One comment is described as following:

“Make it more convenient to rotate using fingers rather than entering degrees”

One of the participants commented that:

“After user defines a parameter like 2 times bigger, the application should update the view without confirmation box.”

Currently, after the user submits a request for updating an image, there will be a wait time for the new image to come from the server. Once the new image has arrived, a message box will pop out to inform the user that a new image is ready to display. Some participants felt annoyed about this message box and suggested removing it so that the new image will be displayed without the need to close the message box.

One of the trial tasks was to change the way the smart phone was held, which would subsequently change the screen direction on the smart phone. Some participants made a suggestion about making the control buttons fit on the landscape direction screen without the need to scroll down in order to click the control button: i.e. control buttons should display on the first page without scrolling down.

5.2.5 Evaluation objectives: results and discussion

In relation to the evaluation objectives stated in section 5.1, the results are discussed in this section.

1. Generate images from server through gateway

In the trial, all participants were able to successfully receive the images generated through the Paraview server on the smart phone.

The questionnaire results showed that all the participants rated “clear” or “very clear” on the displayed image. The mobile application provided a certain size of image box to display the generated image. Therefore, each time the mobile client received an image, the mobile application would automatically display the correct size on this smart phone. This may be an issue when the mobile application is applied on a mobile device with a different screen size. However it is quite easy to change the size of the image box when the application is applied using a different mobile device.

2. Manipulate visualisation on smart phone

The participants were able to use the rotation function to view the object in different directions. They were also able to scale the received visualisation by using the zoom function. The participants could easily change different representations of the visualisation on the smart phone. The resulting visualisations were displayed correctly.

Some of the participants found it easy to use the update functions, while some found them somewhat difficult when using the rotation function to manipulate the subject. This was partly because some participants were not familiar with 3D camera directions. Another reason was that they were not certain about the degree of rotation to input.

3. User interface

The mobile application includes several functions. Most of the participants thought it was easy to use for completing tasks. From the observation and interview results, most of the participants felt the functions on the device were clear to understand and straight forward. A couple of participants made suggestions on how to improve the rotation function button on the smart phone.

Also, the mobile application offered data validation when input was incorrect. The participants found the messages simple to understand and were able to use them to quickly correct input errors.

4. Time issue

In Section 5.2.2, the results show that the participants generally needed more time to update images on the smart phone compared with a desktop computer. It is clear that there is a time gap for the server to transfer the image onto the mobile device through the network. Although mobile tasks took a longer time to complete compared with desktop tasks, despite the mobile network bandwidth, it should also be noted that some participants were not familiar with using smart phones compared with using computer.

5. Expected Output

The trial results and observations showed that all participants were able to provide correct answers for all mobile tasks. In other words, all of the mobile functions were able to implement on the smart phone. Although many participants found the mobile rotation tasks were more difficult, they still were able to receive the required updated image on the smart phone.

5.3 Summary

To evaluate the usability and efficiency of using a smart phone to update a visualisation generated from the Paraview system, a mobile application was provided for the participants in order to complete certain tasks and to compare it with using a desktop computer. All participants were able to successfully complete both mobile and desktop tasks. The participants were able to successfully receive and update the visualisations as required, using the mobile application.

From the trial results, the mobile application was tested out in relation to the objectives described in section 5.2.5. The usability of the mobile application was tested by all participants, with responses fed back using questionnaires and interviews. Although the mobile application demonstrated the concept of using a smart phone to update a visualisation generated from the Paraview system, the participants found some problems with the application's functions that should be improved in the future, such as rotation in real time, landscape layout and use of the confirmation message box.

Chapter 6

Conclusion and future work

This chapter gives a conclusion to this thesis. Based on the thesis objectives and the discussion of the evaluations in chapter 5, this chapter also includes the overall achievements of this thesis. Finally, we suggest future improvements that could be implemented based on the results of the evaluation.

6.1 Conclusion

Mobile devices are being used increasingly in people's lives. Such devices include PDAs and smart phones. Especially for smart phones, besides their calling functionality, new technology and software applications are developing rapidly to make contributions in other areas of communication.

In addition to improved smart phone technology, visualisation systems and computing technology have also developed significantly, allowing visualisations to be viewed on small mobile devices such as smart phones.

Because of the size and mobility of smart phones there are many potential uses of these visualisation applications, however, compared with a desktop computer, mobile devices have some limitations: including limited memory storage, slower performance, and limited graphics hardware. In terms of using visualisation systems on mobile devices, the client-server architecture offers a remote visualisation solution in order to overcome these disadvantages.

Remote visualisation architecture is used widely for lower performance devices to access high performance resources. The concept is simple, but the implementation of the visualisation system and end user input can be complex and time consuming. Since the uses may differ, the server, the client and the visualisation may also vary. As a result, depending on the distribution of the visualisation workload divided between the server and the client, the design of remote visualisation architectures can be quite different.

In this thesis, we have considered smart phone functionality which allows users to interact and view visualisations using a customised visualisation application on a smart phone using data input. The overall objective was to investigate how to simplify generating and updating visualisations using a smart phone and an existing visualisation system. We have developed requirements for a mobile visualisation application in order to generate an appropriate visualisation on a smart phone and to simplify user input at the client end. The mobile application should be able to connect with the server and rendering system.

To minimise the workload of the smart phone, we pointed out that the customised mobile application must be simple, without having to store data and without installing a visualisation system on the device. We also noted that the application must not require users to work with a complex visualisation system on the mobile device. In addition, the solution should allow users to interact with the current visualisation via the server.

In order to meet our requirements, we investigated a gateway approach, acting as middleware to control the interaction between the server and the mobile client (the smart phone). As a proof of concept for using a smart phone to retrieve and update visualisations with an existing visualisation system, the gateway was developed to interact with the server as well as to communicate with the mobile client. The detailed design is described in section 3.5. The gateway has been implemented in Python, with Paraview used for rendering the visualisations. The smart phone application is programmed in VB.NET to test out the concepts in this thesis.

We have evaluated how well the gateway approach met the requirements via two sets of user trials. All participants were able to successfully complete both mobile and desktop tasks. The results are discussed in chapter 5.2. The feedback showed that the gateway approach was successfully implemented in this thesis. There are some suggestions on improving the smart phone's current visualisation functions (we discuss this in the section 6.2). With reference to the overall objectives of the thesis, the achievements are summarised.

- **Application should allow users to create a connection from the mobile device to the visualisation system in an easy way**

We adopted an approach which allowed users to connect with the remote server through a gateway. Users did not need to install the Paraview client mode application on the smart phone. This saved memory storage space on the smart phone.

- **Application should be able to allow users to interact with the visualisation system and update visualisations from a smart phone**

The mobile application we provided allows users to communicate with the render server in the Paraview application via the gateway. Users manipulate the visualisation by passing update commands to the middleware.

From the questionnaire results in the evaluation section, all the users were able to successfully view the image on the smart phone. Users found the existing rotation update functions needed to be improved but were able to update visualisations following instructions.

- **Application should have user friendly interface that can provide data validation and pass useful error messages back to the user**

The mobile application provides automatic data validation. It passed back useful error messages from the gateway when users input invalid data on the smart phone.

From the observation of the user trial, the application successfully checked the input data and generated suitable error messages to alert users.

➤ **Concept may also be applied using similar visualisation systems**

The gateway approach was developed using a thin client mode for remote visualisation. For an existing visualisation system such as Paraview, this prototype offers an alternative to a web-based remote visualisation solution. From the implementation point of view, the gateway concept was successfully accomplished using an existing visualisation application. Therefore, the gateway concept may apply to other visualisation systems when using a smart phone.

6.2 Future work and development

This study provides a different approach for implementing remote visualisations on mobile devices. However there are limitations in the current solution. According to the results and feedback from the user trial evaluation, there is a need for additional scope for improvement. In the future, we will have to consider the following aspects:

➤ **Improvement of existing functions on the device**

Based on users' feedback, there are two major problems regarding the smart phone interface: the rotation function and confirmation message box

The current rotation function confused some users in the trial. Users normally failed to input an appropriate value for degrees to generate the image the first time. They needed to test it a couple of times to guess the input value for rotation, which can be quite annoying. To overcome this issue, a re-design of the rotation function is essential, e.g. add another combo box on the application that allows users to select default angle increments, such as rotate certain degrees (may set up a combo box to select different increments) at each time when a button is clicked to rotate through an angle rather than input the number of degrees.

Currently in the mobile application, after the user submits a request for updating an image, once the new image arrives, a message box pops out to inform the user that a new image is arrived for subsequent display. Since some participants felt annoyed by it, we may automatically make the application close this message box after a couple of seconds without having to ask users to click and close it. In this way, users will still have a message to inform them that the new image has been

displayed, also users will not need to click the close button every time when the new message is ready.

- Allow users to load different data from the mobile device

Since this client application is aimed at testing the use of a smart phone to view and update images using an existing visualisation system, we used a default dataset to render visualisations. In the future, we will need to develop an advanced version to allow users to access more dataset files that are located on the server from the mobile device.

- Support collaborative visualisation

Currently, this study only enabled one smart phone user to view and update visualisations. However with the remote visualisation principle, there could be multiple clients retrieving the visualisation for collaborative updating. In the next development stage, we could adapt the gateway to offer different access points to collaborative clients. Therefore, different users can exchange requests and update visualisations collaboratively

- The extension to other mobile devices

The thesis used a Windows smart phone to test out the gateway concept. However for mobile devices that have other systems, i.e., Android, iOS, there are restrictions on this mobile application. If we want to carry this study forward, it would be necessary to implement the mobile application using software that may be used across multiple platforms.

6.3 Summary

This thesis investigated how to use a smart phone to view and update a rendered image by interacting with an existing visualisation system. A gateway approach was used as middleware based on client-server architecture. It provides a mobile application for a smart phone client, and a gateway to interact between the Paraview server and the mobile client.

In addition, this approach was evaluated using several user trials. The results show that the gateway approach may be used to transfer and update visualisations remotely using a mobile device, and successfully demonstrated using a smart phone to manipulate a visualisation with the Paraview system.

Appendix A

User Trial Mobile Task

Participants Number: _____

Note: Please say when you start or finish a task and “think aloud” whilst you are doing the tasks.

A.1 Mobile Task

1. Create a connection with ParaView (Server details supplied)

2. Rotate the visualisation to show the teapot body.

Please write down the second colour of the **teapot body** (other than Red):

3. Rotate the visualisation to find the lid.

Please write down the **Third colour of the Lid** (other than the two colours above):

4. Make the teapot **two times bigger** than the current view

5. Make the teapot **half** of the current view

6. Put phone into landscape mode

7. Change the representation of the displayed image to “**Wireframe**”

8. Try to go back to the original view of the object

Appendix B

User Trial Desktop Task

Participants Number: _____

Note: Please say when you start or finish a task and “think aloud” whilst you are doing the tasks.

B.1 Desktop tasks:

1. Create a connection with Paraview server (server details supplied)
2. Load the teapot file from the Paraview server (Location: C:\vtk\Data\DesktopTask.pvsm)

3. Rotate the visualisation to find the handle.

Please write down the second major colour of the Handle (other than Blue):

4. Rotate the visualisation to show the teapot body.

Please write down the third colour of the teapot body (count order from lid to bottom):

5. Make the teapot approximately two times bigger than the current view
6. Make the teapot approximately half of the current view
7. Change the representation of the displayed image to “Points”
8. Try to go back to the original view of the object

Appendix C

User Trial Questionnaire

C.1 Interview Questionnaire

Participant Number: _____

Which tasks did you complete first?

- Mobile tasks
- Desktop tasks

Please circle the answer when necessary:

1. Have you used a smart phone before?

- No ==> Go to question 4
- Yes

2. How would you rate your familiarity with using a smart phone?

- Expert
- Average
- Beginner

3. How often do you use your mobile device to download a file:

- Several times a day
- Once a day

- Several times a week
- Once a week
- Less than once a week
- Never

4. Have you used ParaView software before:

- No ==> Go to question 6
- Yes

5. How would you rate your knowledge of ParaView?

- Expert
- Average
- Beginner

6. Is there any visualisation software other than ParaView that you have used for data analysis and visualisation?

- No ==> Go to question 8
- Yes

7. What visualisation software have you used:

Now based on the tasks you have completed earlier, we are going to ask you some questions about those tasks. Please turn to next page.

8. How do you feel about completing the different tasks? Please circle your answer.

Mobile Tasks:	Very			Very		Didn't
	Easy	Easy	Neutral	Difficult	Difficult	
a. Establish network connection	1	2	3	4	5	6
b. Rotating:						
View body	1	2	3	4	5	6
View lid	1	2	3	4	5	6
c. Zoom:						
Zoom in (2 times bigger)	1	2	3	4	5	6
Zoom out (0.5 times smaller)	1	2	3	4	5	6
d. Change display to landscape						
Mode	1	2	3	4	5	6
e. Change Representation	1	2	3	4	5	6
f. Return to initial image (task8)	1	2	3	4	5	6

Desktop Tasks:	Very			Very		Didn't
	Easy	Easy	Neutral	Difficult	Difficult	
a. Establish network connection	1	2	3	4	5	6
b. Rotating:						
View handle	1	2	3	4	5	6
View body	1	2	3	4	5	6
c. Zoom:						
Zoom in	1	2	3	4	5	6
Zoom out	1	2	3	4	5	6
d. Change Representation	1	2	3	4	5	6
e. Return to initial image (task 8)	1	2	3	4	5	6

9. If there's any task you didn't complete in mobile/desktop, please write down why were you unable to complete those tasks?

10. Which **mobile tasks** did you find most difficult. Explain why?

11. Which **desktop tasks** did you find most difficult. Explain why?

12. How do you feel about the clarity of the visualisation displayed on the mobile device?

Very clear Clear Not very clear Not clear at all

1 2 3 4

13. How do you feel about the clarity of the visualisation displayed on the desktop?

Very clear Clear Not very clear Not clear at all

1 2 3 4

14. During the trial, if you got any unexpected results, please describe the unexpected results:

15. What do you like about the mobile application?

16. What do you not like about the mobile application?

17. In your opinion, what are the advantages of using mobile instead of desktop visualisation?

18. In your opinion, what are the disadvantages of using mobile instead of desktop visualisation?

19. If you are away from your work station and need to view the visualisation you worked on, how useful do you think this mobile application would be?

Very useful Useful Not very useful Not useful at all

1 2 3 4

Please tell me your reasons for your selection:

20. Do you have any suggestions on improving the mobile application based on your experiences? If so, please write them down.

Appendix D

User Trial Introduction Session

D.1 Instructions to participants

The purpose of this user trial is to test the usability of using a smart phone to work with an existing visualisation. In order to compare using a mobile device with a desktop computer, this trial will have two sets of tasks for you to complete (the order is randomly chosen):

- Mobile tasks
- Desktop tasks

Once you finish all the tasks, you will be given a questionnaire to complete related to those tasks.

Before you start doing the tasks, I am going to give you a short demonstration and some instructions about the visualisation functions of the mobile device and desktop. Please ask questions along the way if you are not clear about my demonstration.

You are encouraged to “think aloud” while performing the tasks. Aspects that would be useful for you to comment on include:

- Ease of completing different tasks
- Ease of using the smart phone
- Any confusion that you may have

Please do not limit yourself to these.

When you feel you have completed each task, please say so.

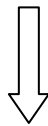
D.2 Mobile phone and web browsing familiarization

We will start the trial with a session in which you can familiarize yourself with the mobile phone. We will use a HTC Pro2 smart phone for this trial.

Please make yourself comfortable with the phone and keypad input.

To change phone to landscape mode:

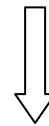
Horizontal mode



© 2009 CBS Interactive

Figure D.1 Horizontal mode

Landscape mode



© 2009 CBS Interactive

Figure D.2 Landscape mode

Retrieve from: http://reviews.cnet.com/2300-6452_7-10001133-4.html?s=0&o=10001133&tag=mncol;thum

When you feel comfortable with the phone and keypad input, please let the researcher know.

D.3 Demonstration

Show user how to input IP address into text box using touch screen keypad


Show user how to change the phone to landscape mode

General talk about the use of functions on mobile device

Show the user some simple rotation (input numbers as degrees in rotation text box)

D.4 Functions on Mobile

D.4.1 Connect to server

Click on  button

Input the given IP address into the text box

Click on  button

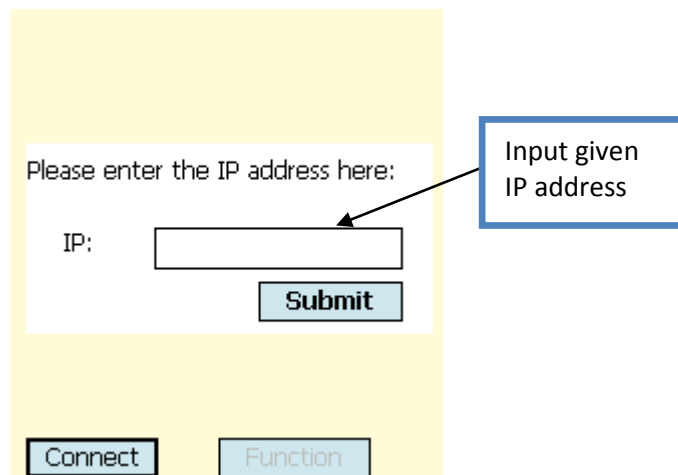



Figure D.3 IP address input for connection

D.4.2 Rotating:

Click on  button to bring out more functions

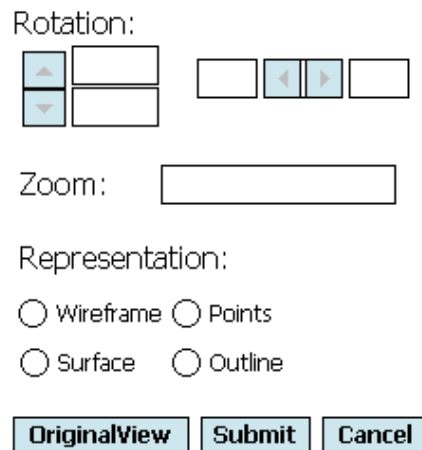


Figure D.4 Rotation function on smart phone

Under “Rotation:” section, choose your rotation direction first. Then input how many degrees you want to rotate in this direction.

For example:

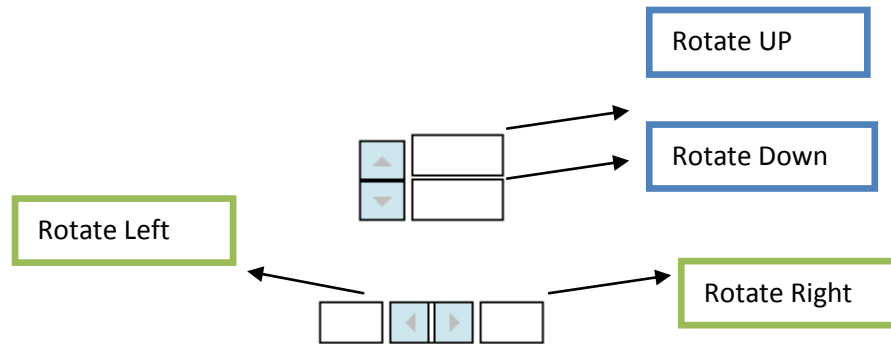


Figure D.5 Rotation function direction



After input the degree numbers, click on **Submit** button, waiting for the new image updated

D.4.3 Representation:

Click on **Function** button to bring out more functions

Under “Representation” section, choose your required representation type

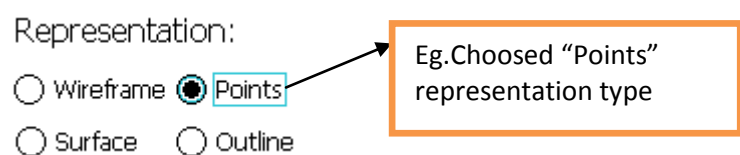


Figure D.6 Representation function on smart phone

Click on button, waiting for the new image updated

D.4.4 Zoom:

Click on button to bring out more functions

Input numbers to zoom in or out.

For example:

Zoom: → *make the object 1.5 times bigger than the current view*

Click on button, waiting for the new image updated

D.4.5 Display first/initial view:

Click on button to bring out more functions

Click on button to reset current view to initial view

D.4.6 Other function:

Click on button under form → bring you back to the visualisation view

Please note you can input as many different requirements as you like in the same time before click on the “Submit” button.

D.5 Desktop Functions

Paraview application running demo:

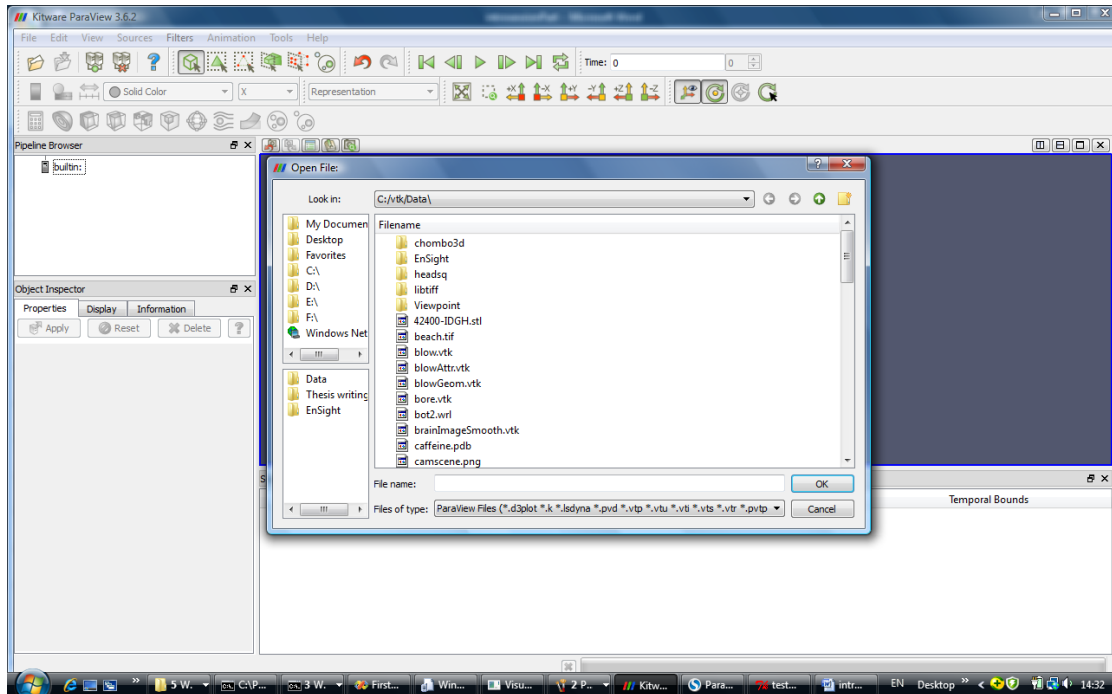



Figure D.7 Desktop Paraview application

D.5.1 Build connection with server:

Click on 'connection' button:  from the toolbar

Choose the existing server 'mb8298.lincoln.ac.nz', and then click on 'Connect'

On the left hand column of the pipeline browser, it should appear like:  `cs://localhost:11111`, which means now this client has successfully connected to the Paraview server.

D.5.2 Load a file:

Go to File → Load State → Select the file you want

Click the  button on the top right hand above the view window



(Note: there are a group of different icons:

D.5.3 Rotating:

Put mouse on the visualisation window area, click and drag around to rotate the visualisation

D.5.4 Representation:

Click on the “newteapot.vtk” view under Pipeline Browser

On the toolbar above the visualisation window, click on the representation tab:



is used to choose different representation type.

D.5.5 Zoom:

Put mouse on the visualisation window area, click and use scroll button on the mouse to change the size of the object view.

D.5.6 Go back to the initial view:

Right click on the file name “newteapot.vtk” under Pipeline Browser, choose “Delete”

Reload the file (see the guide under “Load a file”)

Note: If anything goes wrong or you want to go back to previous view, click on “Undo” button



on the toolbar section.

References

- Applegate, L. M. (1991). Technology support for cooperative work: A framework for studying introduction and assimilation in organizations. *Journal of Organizational Computing and Electronic Commerce*, 1(1), 11-39.
- Bajaj, C., & Cutchin, S. (1999). *Web based collaborative visualization of distributed and parallel simulation*. Retrieved from <http://portal.acm.org/citation.cfm?id=328712.319336>
- Brodie, K. W., Duce, D. A., Gallop, J. R., Walton, J., & Wood, J. D. (2004). Distributed and Collaborative Visualization.
- Charters, S. M. (2010). *Visualization for eResearch: Past, Present and Future*.
- Chittaro, L. (2006). Visualizing information on mobile devices. *Computer*, 39(3), 40-45.
- Diepstraten, J., Görke, M., & Ertl, T. (2004). Remote line rendering for mobile devices.
- Garbow, Z. A., Yuen, D. A., Erlebacher, G., Bollig, E. F., & Kadlec, B. J. (2003). Remote visualization and cluster analysis of 3-D geophysical Data over the internet using off-screen rendering. *Visual Geosciences*, 10-15.
- Grimstead, I. J., Avis, N. J., & Walker, D. W. (2005). *Visualization across the pond: how a wireless PDA can collaborate with million-polygon datasets via 9,000 km of cable*. Retrieved from <http://portal.acm.org/citation.cfm?id=1050498>
- Haber, R. B., & McNabb, D. A. (1990). Visualization idioms: A conceptual model for scientific visualization systems. *Visualization in scientific computing*, 73-74.
- Heer, J. M. (2008). *Supporting Asynchronous Collaboration for Interactive Visualization*. UNIVERSITY OF CALIFORNIA.
- Huang, J., Bue, B., Pattath, A., Ebert, D. S., & Thomas, K. M. (2007). Interactive illustrative rendering on mobile devices. *IEEE Computer Graphics and Applications*, 48-56.
- Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirchner, P., et al. (2002). Chromium: a stream-processing framework for interactive rendering on clusters. *ACM Transactions on Graphics*, 21(3), 693-702.
- Lamberti, F., Zunino, C., Sanna, A., Fiume, A., & Maniezzo, M. (2003). *An accelerated remote graphics architecture for PDAS*. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.1498&rep=rep1&type=pdf>
- Luke, E. J., & Hansen, C. D. (2002). Semotus visum: a flexible remote visualization framework. *IEEE Visualization, 2002. VIS 2002*, 61-68.
- Nadalutti, D., Chittaro, L., & Buttussi, F. (2006). *Rendering of X3D content on mobile devices with OpenGL ES*. Retrieved from <http://portal.acm.org/citation.cfm?id=1122594>
- Park, S., Kim, W., & Ihm, I. (2008). Mobile collaborative medical display system. *Computer methods and programs in biomedicine*, 89(3), 248-260.
- Pea, R. D. (1993). The collaborative visualization project. *Communications of the ACM*, 36(5), 60-63.

- Roberts, J. C. (1993). An overview of rendering from volume data including surface and volume rendering.
- Rodrigues, M. A. F., Barbosa, R. G., & Mendonça, N. C. (2006). *Interactive Mobile 3D Graphics for on-the-go visualization and walkthroughs*. Retrieved from <http://portal.acm.org/citation.cfm?id=1141277.1141516>
- Sørensen K, C. P., Svidt K. (2009). Prototype development of an ICT system to support construction management based on virtual models and RFID. *Information Technology in Construction (ITcon)*, 14(Special Issue Next Generation Construction IT: Technology Foresight, Future Studies, Roadmapping, and Scenario Planning), 263-288.
- Schroeder, W., Martin, K., & Lorensen, B. (2002). *The Visualization Toolkit An Object-Oriented Approach To 3D Graphics*, Kitware. Inc.
- Squillacote, A. H. (2007). *The ParaView Guide: A Parallel Visualization Application*: Kitware.
- Watt, A., & Policarpo, F. (1998). *The computer image*: ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.
- Yuen, D. A., Garbow, Z. A., & Erlebacher, G. (2004). Remote data analysis, visualization and problem solving environment (PSE) based on wavelets in the geosciences. *Visual Geosciences*, 9(1), 29-38.
- Zhou, H., Qu, H., Wu, Y., & Chan, M. (2006). Volume visualization on mobile devices. *National Taiwan University Press: Taipei*, 76-84.