Utilising the Grid for Augmented Reality

Chris J. Hughes

A Thesis presented for the degree of

Doctor of Philosophy

PRIFYSGOL
BANGOR
UNIVERSITY

1884

High Performance Visualization and Medical Graphics
School of Computer Science
Bangor University
UK

January 2008

Utilising the Grid for Augmented Reality

Chris J Hughes

Submitted for the degree of Doctor of Philosophy

January 2008

Abstract

Traditionally registration and tracking within Augmented Reality (AR) applications have been built around specific markers which have been added into the user's viewpoint and allow for their position to be tracked and their orientation to be estimated in real-time. All attempts to implement AR without specific markers have increased the computational requirements and some information about the environment is still needed in order to match the registration between the real world and the virtual artifacts. This thesis describes a novel method that not only provides a generic platform for AR but also seamlessly deploys High Performance Computing (HPC) resources to deal with the additional computational load, as part of the distributed High Performance Visualization (HPV) pipeline used to render the virtual artifacts. The developed AR framework is then applied to a real world application of a marker-less AR interface for Transcranial Magnetic Stimulation (TMS), named BART (*Bangor Augmented Reality for TMS*).

Three prototypes of BART are presented, along with a discussion of the subsequent limitations and solutions of each. First by using a proprietary tracking system it is possible to achieve accurate tracking, but with the limitations of having to use bold markers and being unable to render the virtual artifacts in real time. Second, BART v2 implements a novel tracking system using computer vision techniques. Repeatable feature points are extracted from the users view point to build a description of the object or plane that the virtual artifact is aligned with. Then as each frame is updated we use the changing position of the feature points to estimate how the object has moved. Third, the e-Viz framework is used to autonomously deploy HPV resources to ensure that the virtual objects are rendered in real-time. e-Viz also enables the allocation of remote *High Performance Computing* (HPC) resources to handle the computational requirements of the object tracking and pose estimation.

Declaration

The material in this thesis has not been previously submitted for a degree or diploma in any university. To the best of my knowledge this thesis contains no material previously published or written by another person except where due acknowledgement is made in the thesis itself.

Chris Hughes

Date: 31st January 2008

Acknowledgements

- Prof. Nigel John, for without his support, ideas, encouragement and his understanding of my creative interpretations of his deadlines, this Thesis would never had existed.
- Dr. Franck Vidal, for not only having managed to share an office with me for over 4 years, but mainly for the contribution of the word 'instead' which he has kindly donated to page 80.
- Dr. Ade Fewings for pathing the way to > 4 year PhD submissions. Also to everyone else in the Graphics Group at Bangor University.
- Jason Lauder and Bob Rafal from the School of Psychology, Bangor University, for allowing us access to their TMS laboratory and for supplying data from past experiments.
- The e-Viz teams at University of Manchester (Dr. John Brooke and Mark Riding), University of Leeds (Prof. Ken Brodlie and Dr. Jason Wood) and Swansea University (Prof. Min Chen, Dr. Mark Jones, David Chisnall and Nicolas Roard) for their help and support.
- Ian Grimstead for his initial encouragement towards research in the field of Visualization.

- My Mum for encouraging me to follow my dreams and for not being dissapointed when I turned out to be an academic.

- Chris Headleand and Sam Godwin, for giving me a much needed outlet for my built up frustration and aggression by helping to bring the sport of Canoe Polo to Bangor. Also to the many people who have folloId in my footsteps to ensure there is always a game to play.

- The many authors and artists who have influenced me over the last 4 years, particularly: Douglas Adams, Robert Rankin, Mark Thomas, Tenacious D.

- This research is supported by the EPSRC within the scope of the project: ”An Enhanced Environment for Enabling Visual Supercomputing” (GR/S46567/01).

List of Abbreviations

AG Access Grid

ANL Argonne National Laboratory

API Application Programming Interface

AR Augmented Reality

ARToolkit Augmented Reality Toolkit

BART Bangor AR for TMS

CAVE Cave Automatic Virtual Environment

CoG Java Commodity Grid Kit

CRC Cyclic Redundancy Check

CSAR Computer Services for Academic Research

DART Designers Augmented Reality Toolkit

EPSRC Engineering and Physical Sciences Research Council

fMRI functional Magnetic Resonance Imaging

FPS Frames Per Second

GGF Global Grid Forum

GPU Graphics Processing Unit

GRAM Grid Resource Allocation and Management

GSI Grid Security Infrastructure

GUI Graphical User Interface

HIT Lab Human Interface Technology Lab

LCD Liquid Crystal Display

HMD Head Mounted Display

HPC High Performance Computing

HPV High Performance Visualization

ICENI Imperial College e-Science Networked Infrastructure

IPG NASA's Information PoIr Grid

IR InfraRed

JPEG Joint Photographic Experts Group

MR Mixed Reality

MVE Modular Visualization Environments

OGSA Open Grid Services Architecture

OGSI The Open Grid Services Infrastructure

OpenCV Open Source Computer Vision Library (Intel)

osgART open scene graph ARToolkit

PDA Personal Digital Assistant

PSE Problem Solving Environment

SDK Software Development Kit

SGS Steering Grid Service

skML skm Markup Language

SOA Service Orientated Architecture

SOAP Simple Object Access Protocol

TMS Transcranial Magnetic Stimulation

UnICoRe Uniform Interface to Computing Resources

V4L Video for Linux

VE Virtual Environment

ViPar Visualization in Parallel

VR Virtual Reality

VRML Virtual Reality Modelling Language

VTK Visualization Toolkit

WSRF Ib Services Resource Framework

WSDL Ib Services Description Language

XML eXtensible Markup Language

Contents

List of Figures

Chapter 1

Introduction

(a) (b)

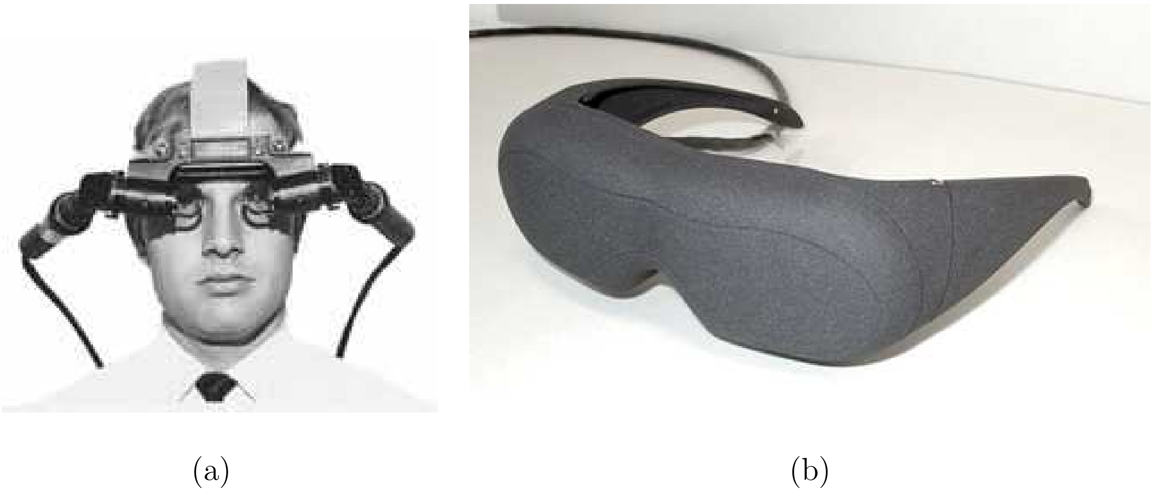
Figure 1.1: Early work at (a) MIT and (b) Bangor University, showing how users were able to interact with the early *Computer Graphics* applications.

Since the Term *Computer Graphics* was first coined in the early 1960's by William Fetter to describe his work at Boeing [1], there has been a constant challenge not only to improve the quality of the computer graphics but also to find more natural interfaces between the *Computer Graphics* image and the real-world. Initially cathode ray-tube (CRT) monitors were the only technology that was available to display the computer output, however as computing resources became more powerful allowing *Computer Graphics* to improve, there was also a requirement to improve the methods by which users view the images.

Even in the early days it was clear how powerful the computer graphics could be. Figure 1.1, gives examples of some of the first computer graphics research being done, both at (a) MIT and at (b) Bangor University. Work done in 1965 by Ivan Sutherland produced the first Head Mounted Display (HMD) as a 'window into a virtual world', allowing the users to look into an environment that was completely computer generated and although the *Computer Graphics* were extremely basic it was a major milestone in the work towards immersive visualization environments [2]. However it was not until 1989 when Jaron Lanier used the phrase *Virtual Reality (VR)* and created the first commercial business around virtual worlds [3].

Twenty years later researchers are still working towards that goal. Computational resources are much faster, most people have watches with more processing power than was available to them in the 1960's and over 67% of all homes have a computer with the capabilities to generate real-time, photo-realistic environments. Ivan Sutherland's CRT based HMD has now been replaced by ultra thin LCD displays, but work is still going on to develop better and more natural ways to allow the user to understand the computer graphics.

During the 1990's, Tom Caudell, another developer at Boeing, suggested the idea of *Augmented Reality (AR)*, allowing the user to remain present in the real world but also to use an HMD to supplement the real world with virtual graphics and information which could be used to help workers at Boeing assemble cables into aircraft [4]. In 1994 Ronald Azuma took this work forward defining three requirements



(a) (b)

Figure 1.2: (a) 1966: Ivan Sutherland's CRT based HMD (b) Modern day: A typical HMD with stereo capabilities up to 800x600 resolution. Commercially available for approx $5000.

for AR: a) the real and virtual world must be combined, b)the environment must be interactive whilst maintaining real-time performance (i.e. more than 15 frames per second) and c) the *Computer Graphics* must be registered in 3D [5].

There have now been many successful applications which use AR, however it is generally accepted that all of them require specifically placed, bold markers to be placed within the environment to enable the software to easily align the virtual objects with the real world. This Thesis presents a solution which enables virtual artifacts to be overlaid into the real world view and tracked using a series of feature points rather than bold markers. The developed AR framework is the applied to the real-world application of a markerless AR interface for Transcranial Magnetic Stimulation (TMS). Furthermore the solution is extended to allow the computer Graphics to be rendered using remote *High Performance Visualization* resources, in order to meet the requirements of our exemplar application.

1.1 Hypothesis

The hypothesis is that it is possible to create an interactive Augmented Reality interface, using feature points for registration. It will also be capable of providing high performance visualizations by seamlessly deploying processing power from a remote, possibly Grid-enabled, high performance computing resource.

The framework developed will be applied to Transcranial Magnetic Stimulation.

1.2 e-Viz Project

The e-Viz project was funded by the EPSRC between November 2003 and February 2007 to provide a solution to the *Grand Challenge* problems which require High Performance Visualization (HPV) resources. It was a collaborative project involving computer science groups at Bangor, Leeds, Manchester and Swansea Universities. By exploiting Grid technologies, e-Viz provides a generic flexible infrastructure for remote visualization allowing the rendering of large datasets whilst providing real-time, high resolution performance. It does this using a combination of intelligent scheduling for new rendering pipelines and the monitoring and optimisation of running pipelines, all based on information held in a knowledge base. This provides an adaptive visualization service that provides rendered graphics reliably without the application or user even being aware of what resources are being used. An overview of the e-Viz framework is given in chapter 2.

The e-Viz project has demonstrated how visualization can be seamlessly integrated into a Grid infrastructure and the application of autonomic computing strategies to visualization tasks - this is the first time that such a strategy has been proposed. Bangor's contribution to the e-Viz project included the study of the performance of distributed graphics pipelines in a visualization architecture built on commodity PC hardware. This reflects the trend of incorporating such architectures in Grid computing, in place of supercomputers. Furthermore, the development of an augmented reality interface for transcranial magnetic stimulation using high performance computing resources, was created as an exemplar application for the e-Viz framework. The work of which is included as part of this thesis and is explained on more detail in Chapter 6.

1.3 Motivations of this work

Augmented Reality (AR) provides the user with the ability to visualize and project 3D data or other information into their natural environment. It also provides the user with an intuitive method of interacting with the data in a way that has not previously been available. By providing the user with an egocentric view of the visualized data allows them to use their natural spatial understanding and to gain a sense of their presence in the real world, augmented with virtual artifacts.

The use of grid enabled visualization resources, to render the virtual artifacts, allows the AR environment to be generated by a workstation that would not normally have the required processing capabilities. Combining this with the possibility to capture the user's viewpoint using a cheap proprietary web-cam allows the AR applications to be brought to the user's home and run on their average, non specialist, desktop computer. Other suitable platforms for running AR applications could include a Personal Digital Assistant's (PDA's) or some other embedded device such as a mobile phone, many of which have camera facilities built in.

The ability to visualize and interact with an environment that has been augmented with complex virtual artifacts could have a great impact upon many applications. Some examples of these include:

Medical: During medical diagnostics or surgery, previously derived tomography could be overlaid over the patient allowing the clinician to ‘see’ inside the patient.

Manufacturing and Assembly: Assembly instructions could be placed into the field of view of the assembly worker.

Visualization of Architecture: Destroyed historic buildings could be virtually rebuilt, or planned construction projects be simulated.

Collaboration: Meetings could combine both real and virtual participants. Also a single 3D artifact could be explored by user’s from multiple locations.

Entertainment: Virtual Objects could be presented in museums and exhibitions

Several AR applications exist that use specific markers for registering the virtual artifacts with the real world. However this thesis presents a solution which uses information extracted from the user’s view in order to track and align the virtual object. The performance of most of the existing marker-less solutions are still beyond the demands of real-world AR applications, in terms of speed, accuracy and flexibility. The user’s viewpoint must be freely movable, which makes the movement of the camera unpredictable. The 3D pose of the object must also be calculated reliably regardless of environmental changes. Therefore any successful solution must satisfy the following criteria:

- Must be easy to calibrate offline which real world objects are being tracked.
- Needs to be capable of automatically reinitializing should the alignment drift or an error occur.
- Needs to extract feature points reliably and repeatably.
- Must have a computational cost that can be satisfied in real-time.
- Must provide an accurate alignment of the virtual and real objects.
- Must work in unconstrained environments, such as changing levels of light.
- must be able to be adapted for each required application.

The main challenges of this research include:

- Implementing a robust, repeatable feature point detector.
- Registering extracted feature points to a calibrated reference frame.
- Implementing a pose estimation algorithm in order to reconstruct the position and orientation of the tracked object.
- Utilise the e-Viz framework to seamlessly deploy the computational load of the AR application to a remote Grid enabled resource.
- Attempt to achieve an Autonomic AR environment with support for Self-configuration, Self-healing, Self-optimization and Self-protection.

1.4 Contributions of this research

This research makes a number of contributions to the current state of the art in *Augmented Reality* (AR) and the use of grid enabled visualization technologies, allowing more computationally intensive AR environments to be generated. Specifically the contributions include:

- A novel AR interface to the *Transcranial Magnetic Stimulation* (TMS) using a *Polaris Optical Tracking System* was developed. This allows the operator to identify and align the TMS tools with the required regions of interest on the subject's cranium, whilst working in a much more natural environment. The software, *Bangor Augmented Reality for TMS* (BART) allows the operator to see the actual rendering of the subjects cranium whilst interacting with the subject rather than focusing at a computer monitor.

- A new framework for tracking, using only a cheap webcam rather than an expensive proprietary tracking system was developed. By extracting feature (or corner) points from the user's viewpoint the software is able to track the position and pose of the real world object that was used for reference. A Haar-classifier technique was combined with a POSIT algorithm to track the position of the moving object and calculate its change in orientation and rotation within set error bounds.

- In order to use BART in real-time with large volume datasets, the e-Viz framework was successfully used to transparently allocate a remote *High Performance Visualization* (HPV) resource to remotely render the virtual artifacts.

- To satisfy the large computational requirements of the BART software it was finally distributed as part of the visualization pipeline. The e-Viz framework was used to allocate remote *High Performance Computing* (HPC) resources.

- During this research a significant contribution was made to the development and testing of the e-Viz framework, a project which was rated as tending to outstanding by the EPSRC reviewers. Specifically, contributions were made to the following components of e-Viz included:

- Contribution to the overall system design.
- Development and maintenance of the e-Viz broker and database which ran on a machine based at Bangor.
- Development of an e-Viz wrapper for openDX, allowing it to be used as a remote visualization software resource.
- Install and testing of the e-Viz software during development.

1.5 Publications

The following publications where contributed to during this research project:

1.5.1 Journal Publications:

(1) K Brodlie, J Brooke, M Chen, D Chisnall, A Fewings, **C Hughes**, N.W. John, M Jones, M Riding, and N Roard, 'Visual Supercomputing - Technologies, Applications and Challenges', Computer Graphics Forum, Number 24, Issue 2, pp. 217-245, 2005.

1.5.2 Publications in refereed conference Proceedings:

(2) K.W. Brodlie, J. Brooke, M. Chen, D. Chisnall, **C. Hughes**, N.W. John, M.W. Jones, M. Riding, N. Roard, M. Turner, J.D.Wood, 'Adaptive Infrastructure for Visual Computing', Proceedings of Theory and Practice of Computer Graphics, Bangor, pp. 147-156, ISBN 978-3-905673-63-0, 2007.

(3) **C. J. Hughes**, N. W. John, 'A generic approach to High Performance Visualization enabled Augmented Reality', Proceedings of Theory and Practice of Computer Graphics, Bangor, pp. 181-186, ISBN 978-3-905673-63-0, 2007.

(4) **Chris Hughes** and Nigel W. John, 'A flexible infrastructure for delivering Augmented Reality enabled Transcranial Magnetic Stimulation', Proceedings of Medicine Meets Virtual Reality 14, Long Beach, California, pp. 219-224, IOS Press, ISBN 1-58603-583-5, 2006.

(5) Mark Riding, Jason Wood, Ken Brodlie, John Brooke, Min Chen, David Chisnall, **Chris Hughes**, Nigel W. John, Mark W. Jones, Nicolas Roard, 'e-Viz: Towards an Integrated Framework for High Performance Visualization', Proceedings of the UK e-Science All Hands Meeting, EPSRC, pp. 1026-1032, ISBN 1-904425-53-4, 2005.

(6) K. Brodlie, J. Brooke, M. Chen, D. Chisnall, A. Fewings, **C. Hughes**, N.W. John, M. Jones, M. Riding, and N. Roard, 'Visual Supercomputing - Technologies, Applications and Challenges', STAR Report, Eurographics, Grenoble, France, 2004.

1.5.3 Refereed Poster Presentations:

(7) K. W. Brodlie, J. Brooke, M. Chen, D. Chisnall, **C. J. Hughes**, N. W. John, M. W. Jones, M. Riding, N. Roard, M. J. Turner and J. Wood, 'A Framework for Adaptive Visualization', Poster presentation for IEEE Visualization, Baltimore, Maryland, USA, 2006. Poster presented by Hughes.

(8) **Chris J Hughes**, Nigel W. John and M. Riding, 'A generic approach to High Performance Visualization enabled Augmented Reality', Poster presentation

at UK e-Science Programme All Hands Meeting, Nottingham, 2006. Poster presented by Hughes.

1.5.4 Summary of contribution to each publication:

Paper (1) was a revision of Paper (6) which was originally drafted as the literature review for this thesis. It was moderately redeveloped, as a state-of-the-art report with contributions from the e-Viz consortium. Paper (2) summarises the final results of the e-Viz project. Paper (3) detailed the BART v3 software, as shown in Chapter 6, and how it was integrated to use the e-Viz framework. Paper (4) was based upon the first version of BART as described in Chapter 4. Paper (5) described the design of the e-Viz system, including details about the Broker web service.

The Poster (7) was prepared and presented by Bangor, on behalf of the e-Viz consortium and submitted as a poster to IEEE Visualization, describing e-Viz and how it aimed towards adaptive visualization. Poster (8) summarized the early work towards BART v3, as shown in Chapter 6.

Chapter 2

Background and Literature Review

2.1 Virtual Environments

Virtual Environments represent a major technical drive in computer graphics and visualisation, and have helped push a range of hardware and software technologies forward. A Virtual Environment (VE) gives the user a feeling of being inside a computer generated environment with a sense of spatial presence and often coupled with physical presence. For many visualisation applications, Virtual Environments can provide user's with realistic experience in interrogating, navigating within, feeling and manipulating data via its visual representation.

The visual display can be broken down into three different types of immersive display. These three types are described below:

2.1.1 Non-Immersive Displays

The most basic method of stimulation is to use a computer display to show the user graphics information, allowing the user to view the the virtual environment through a window. Generally these environments allow the user to interact using a mouse or a keyboard. However using an ordinary computer display has many limitations and the user would only see a flat image and do not give the user a true feeling of being immersed within the virtual environment. To try and extend upon this scientists have developed several ways to send a different image to each eye and thus give the user a 3D perspective of the image.

2.1.2 Semi-Immersive Displays

One popular method is to use active-stereo and Shutter Glasses, as shown in Figure 2.1(a). With this technology an ordinary computer monitor can be used and allow several people to view the visualization at a time. These work by using LCD panels to alternatively block the light to each of the user's eyes. This can then be synchronized to the monitor which can display different projections for each eye by switching between the images very quickly [6]. These allows the user to get a true grasp of the 3D environment and although the user can still see the real world around them, gives them a more realistic view of the objects and environment into which they are looking. There are several commercial versions of the Shutter Glasses from companies such as CrystalEyes [7] and StereoGraphics [8].

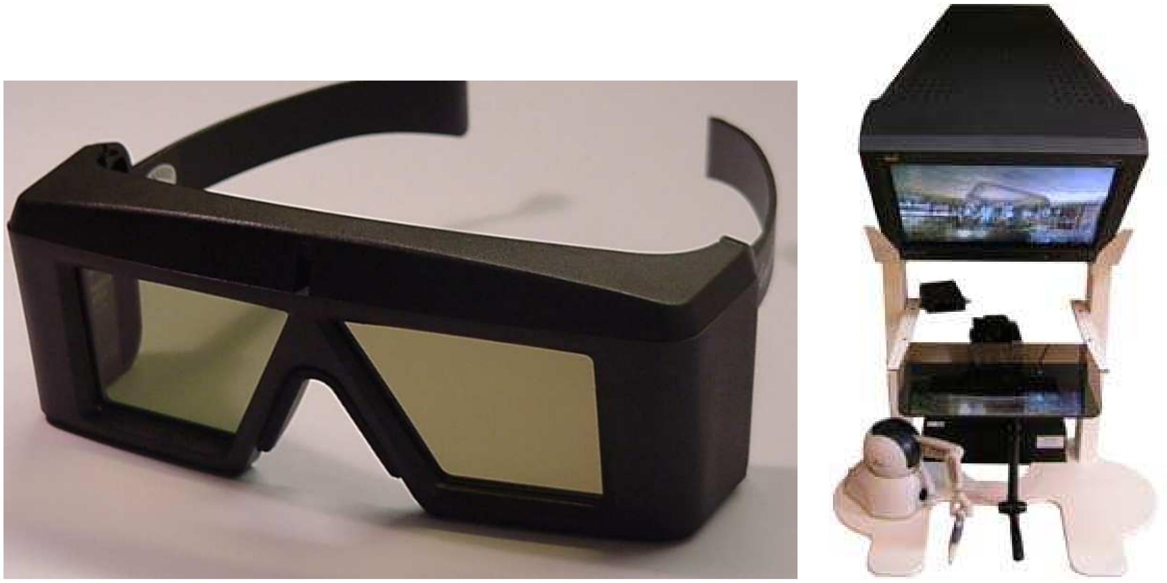
Shutter Glasses although very effective for small groups of people, are not suitable for larger or public events due to their cost. An inexpensive alternative is to use a passive-stereo alternative [9]. In this case each viewer is given a disposable pair of glasses containing two oppositely polarized filters, which can be produced very cheaply. The image is projected from two digital projectors with their image filtered to match the separate polarization filters in the viewer's glasses. This means that each eye can only see the image from one of the projectors and therefore the stereo image can be produced by sending a different projection to each eye.

Auto-stereoscopic displays are being developed to enable the user to see into the virtual environment in 3D without the need for Shutter Glasses, such as the SeeReal display from inition [10]. However many of these displays can be difficult to set up successfully and require the user to be sit in a specific position, making it difficult for collaborative use.

Another example of semi-immersive displays is the Immersive Workbench from Sensegraphics [11]. They have developed an environment which uses a standard computer display, which is reflected from a semi-transparent mirror. This allows the user to look down into the virtual environment through the mirror using Shutter Glasses. This allows the user to not only see the 3D environment, but also leaves the real workspace within the environment empty for other tools to be used, such as Haptic devices which allow the user to touch the objects within the environment. The Immersive Workbench is available in a number of different formats, ranging from the desktop version, as shown in Figure 2.1(b) using a CRT monitor, up to a collaborative version using a 3D digital projector to provide a much bigger working environment.

2.1.3 Fully-Immersive Displays

The first method to allow a user to be fully immersed in a virtual environment was to mount a visual display in front of each eye using a headset design to form a *Head Mounted Display* (HMD). This was limited because only one user would be able to



(a) (b)

Figure 2.1: An example of (a) a pair of Shutter Glasses from CrystalEyes and (b) a desktop version of the Immersive Workbench from Sensegraphics.

see the visualization at once, but it did ensure that the user would only be able to see the computer environment. This provides a number of other limitations such as the headsets were generally uncomfortable due to their size and weight. They also the user requires some form of cabled connection to the computer which could be cumbersome [12].

The University of Illinois at Chicago developed the *Cave Automatic Virtual Environment* (CAVE) in 1992 [13]. Its main aim was to address the challenges of providing a visualization tool which allowed multiple user's to experience the environment simultaneously. The CAVE used a series of large scale digital projectors to surround the user's, enabling the environment around the user to be controlled by the computer system, as ilustrated in Figure 2.2. The CAVE became an enormously successful interface to virtual environments and several commercial systems are available. Fakespace Systems [14] are one of the leading suppliers of immersive displays.

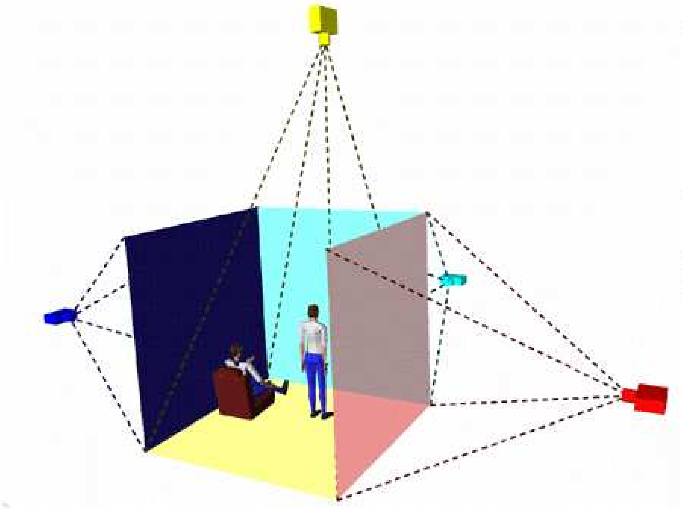


Figure 2.2: The basic set up of the *Cave Automatic Virtual Environment* (CAVE)

2.2 Augmented Reality

Augmented Reality (AR), is an extension of existing Virtual Environments (VE) which completely immerse a user inside a computer generated environment. By contrast, AR allows the user to see the real world, whilst supplementing it with virtual objects that are superimposed within the real world [5].

Milgram and Kishino's continuum, shown in Figure 2.3, illustrates the difference between reality and virtuality with several steps between them. They define any step between the real world and a completely virtual environment as Mixed Reality (MR). In Augmented Virtuality real world views or objects are inserted into a virtual scene, rather than in AR where virtual objects are inserted into real world scene.

There are many potential areas currently being explored as possible AR applica-

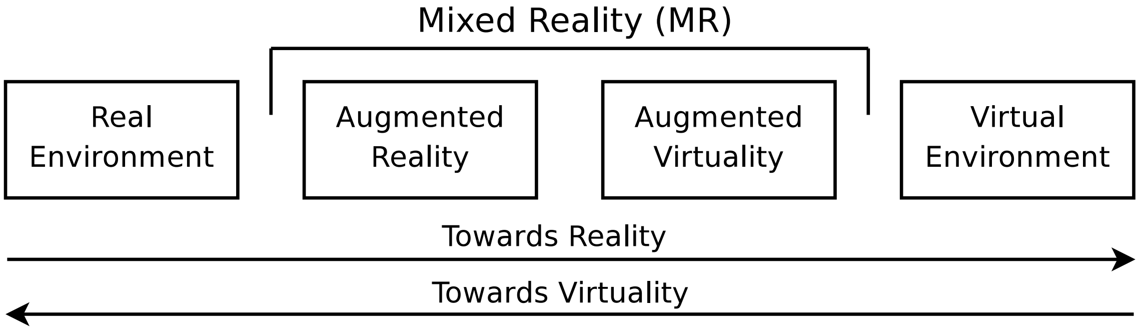


Figure 2.3: Milgram and Kishino's reality-virtuality continuum.

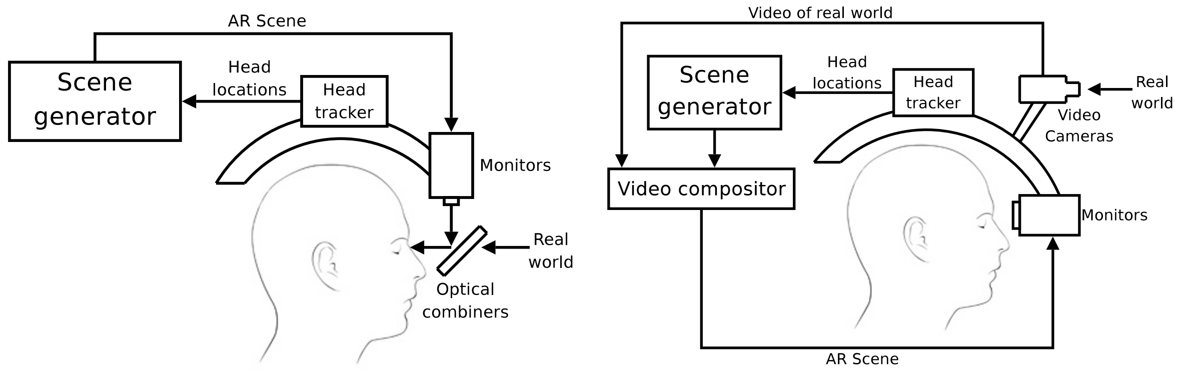
tions, including: medical visualization, maintenance and repair, annotation, robot path planning, entertainment and aircraft navigation [5].

2.2.1 Displays

Most AR technologies have been based upon the use of some form of transparent display which is positioned between the real world and the eyes of the user [15]. In order to align the computer graphics with the physical reality, cameras are used to track the movements of the user's vision and allow the graphics to be realigned [16]. It is also possible for a collaborative AR in which several users can be tracked and see the same virtual objects from different perspectives [17].

This can be done most simply using a HMD which is designed to allow the user to 'see-through' an optical combiner and therefore to be able to see the real world as well as the computer graphics as shown in Figure 2.4(a). Other HMD solutions include video see-through HMD's which use dual cameras (to capture the real world in stereo) as shown in Figure 2.4(b). The video streams captured by these cameras is then composited with the computer graphics to generate the user's view.

Volumetric Displays attempt to produce 3D images within a free space. This



(a)

(b)

Figure 2.4: Comparison of (a) Optical see-through HMD and (b) Video see-through HMD.

is usually done by the emission of a light source such as a laser into 3D space. Actuality Systems Inc, produce an auto-stereoscopic device which allows users to walk around and see a life size 3D image. It works by projecting a series of 2D images onto a rotating screen. Although this allows many people to look at the virtual objects simultaneously, it is very restrictive as the 3D objects can not be positioned arbitrarily into the real world.

2.2.2 Applications

The most basic method AR implementation is to overlay computer graphics onto a 2D tabletop surface. Rekimoto et al [18] developed an InfoTable which combined a series of cameras to identify real objects that were placed on the tabletop, and an LCD projector which was used to append the known objects with useful information.

Several AR techniques have now been shown to add value to the information available to doctors in the medical world. Magnetic Resonance Imaging (MRI) and Computed Tomography scans (CT) can be used to build 3D datasets of a patient.

This data can then be rendered in real time and overlaid onto the patient allowing the doctor virtually to see inside the patient [19].

The Eurographics Association, sponsor an annual medical prize, acknowledging research utilizing computer graphics within the medical field [20]. In 2003 the prize went to an AR application for Liver Surgery Planning, which utilized a Personal Interaction Panel (PIP) and a tracked pen to allow doctors to examine a patients liver. The PIP was used to allow doctors to specify cross sections of the liver that they wanted to examine.

Larose used a Tile system to create a PIP which was a two handed pen and pad interface into AR applications as part of the Studierstube AR Project [17] This panel allowed user's to interact with virtual controls overlaid onto the panel.

2.3 High Performance Computing and the Computational Grid

'High Performance Computing (HPC) involves the use of parallel computing systems to solve computationally intensive applications' [21].

A Grand Challenge problem is a complex problem to which no ideal solution has yet been found. Grand Challenges are all demonstrably hard to solve, but are believed to have a solution and have a significant social or economic impact. Within science and engineering many of the existing Grand Challenge problems have not been solved due to computational requirements that cannot be satisfied on single processor machine. In order to solve this, the idea of parallelism has been introduced which allows a task to be split into smaller parts which can be processed in parallel on multiple processors.

There is a lot of research into the field of parallel algorithms, with a particular focus on developing algorithms which can provide scalability to perform efficiently depending upon the available resources.

Parallel architectures have also become very popular due to the efficiency that can be achieved at low cost. For example, it is possible to achieve the same level of processing power by using a number of cheap, desktop computer processors as you could from one expensive specialist processor [22].

There is no completely satisfactory way to characterize the different types of parallel system. Flynn [23] devised a taxonomy that is still the most popular and widely used today. Streams of information are used to as the basis of classification. The streams of data that are received by a processor can be separated into two separate groups; instructions and data. Flynn's taxonomy classifies each node in a parallel system according to whether it has one or more streams for each type of information.

2.3.1 Single Instruction, Single Data Stream (SISD)

This is the traditional uniprocessor model, where only one instruction can be executed at a time. This means that it is not possible to achieve parallelism, although it can closely be emulated by multitasking. This means that each process shares the processor usually on a time share basis.

2.3.2 Single Instruction, Multiple Data Streams (SIMD)

This model is used to describe many processors each executing the same instructions in lock step, but on different sets of data. This means that there is only one instruction counter and each processor performs the same operation on its separate data in sync. This global synchronization is generally performed by hardware. SIMD machines are particularly useful for processing vectors or images, where the data can be partitioned into separate blocks.

2.3.3 Multiple Instruction, Single Data Stream (MISD)

There are very few machines which would fit into this category, certainly none which have been particularly successful or had any impact on computer science. To construct an MIMD system several instructions streams would need to operate on the same data simultaneously. One example of an MISD machine is the experimental Carnegie-Mellon computer from 1971.

2.3.4 Multiple Instruction, Multiple Data Streams (MIMD)

This is the most general and most powerful model for high performance computing. Execution on each processor can be either synchronous or asynchronous, deterministic or non-deterministic, running its own individual set of instructions on its own set of data.

It is possible to construct an MIMD machine using simple of the shelf processors, which are cheap and easily available. There are also a variety of remote message passing software systems available to allow the use of workstations on a network as MIMD systems.

2.3.5 Other Taxonomies

There are several weaknesses in the structure of Flynns taxonomy. The main problem occurs in the MIMD category. With Flynns taxonomy all of the MIMD systems are categorized together with no regard for how they are actually interconnected, and how they are connected to the shared memory. Since these characteristics can have such a dramatic effect on the performance of the system it would be more suitable to classify them differently.

A popular solution to this has been to extend the SIMD category to Single Program Multiple Data stream (SPMD) as it allow processing similar to SIMD on MIMD hardware. It is commonly used for trivially parallel problems such as queueing [24].

2.3.6 Example of a Parallel System

The University of California, Berkeley has a research department which is looking into more advanced techniques for analysing radio transmissions received by the Arecibo telescope in Puerto Rico. Their main project is called 'The Search for Extraterrestrial Intelligence' (SETI) and involves a number of mathematical functions which look for correlations in the received radio data.

Each day about 36Gb of data is recorded and has to be analysed. In order to do this however would require a large amount of very expensive processors. The team at Berkeley realized how much processor time was being wasted by privately owned computers sitting idle and realized that all of these computers could be utilized in the search for extraterrestrial transmissions.

In 1996 the idea for SETI@Home was first conceived that a screen saver could be released to the public which would be able to request small chunks of the data and perform the analysis on the data when the computer was idle. Finally in 1999 the software was released to the public. A lot of media attention launched the project and soon many computer user's where keen to do their bit and to be involved.

The SETI@Home client works by requesting a data segment from the server hosted at Berkeley. The server then sends back 0.25Mb of data to the client and keeps a record of where the processing is taking place. Once the data segment is processed which can take several hours, the result is returned to the server which correlates all of the results back together. If for any reason the server doesn't receive a reply within a specified period the segment of data is simply sent to a different client.

The SETI@Home project is an example of how supercomputer power can be achieved without the need to buy expensive mainframe computers. Now with nearly 4.5 million user's there is the potential for millions of nodes to be running in parallel. To put this into perspective the SETI@Home website tells us that the most powerful computer, IBMs ASCI White, is rated at 12 TeraFLOPS and costs $110 million. SETI@home currently gets about 15 TeraFLOPs and has cost $500K so far [25] [26].

2.4 High Performance Visualization

High Performance Visualization (HPV) is the combination of the very latest visualization techniques along with the utilization of High Performance Computing (HPC) resources.

HPV generally involves high resolution graphics, large quantities of data and computationally-intensive tasks. HPV tasks often make use of distributed data and extensive network communication. Typically HPV tasks can be seen as 'complex feedback processes, involving data collection, visualization design, task parallelisation, immersive visual display and interfacing with the corresponding data generator such as a simulation engine' [27].

2.4.1 Visualization

The basic aim of visualization is to convert a subset of data into a more perceptually understandable form by using computer graphics to represent the data.

The technique of visualization was around long before computers. Early scientists where forced to record their data using hand drawn images and diagrams to represent the information. This would have been a painstaking task and meant that early researchers had to be accurate artists as well as scientists [28].

With the progresses in technology users are now able to generate vast quantities of numerical data during our work or research. This can be achieved in many different ways, including: computational simulations and actual measured data.

In order to implement visualization solutions it is necessary to find a method of presenting the original data into a graphical way that will be useful to the researcher and there are many trade-off's that will have to be made to find the most efficient method. Most of the current limitations with visualization are caused by the physical limitations of the computer systems being used. For example memory limitations may mean that the computational process will only be able to handle a small subset of the data at one time or the processing capabilities may mean that the computer system isn't capable of processing the graphics in real time.

There are a great many different solutions to this problem, which the combination of visualization with HPC is starting to improve but not yet completely overcome. Different solutions to improve this include batch processing, where complex data which can not be rendered in real time can be rendered in advance, or allowing the researcher to work on low resolution data and when an area of interest has been identified then use high resolution data for this region. It may also be possible for the researcher to manipulate the low resolution data in real-time whilst recording his movements. Then this path could be retraced later and an animation rendered over a longer period of time.

When the entire view of a dataset is rendered it is known as a qualitative overview but when a specific subset of the data is being represented then it is known as a quantitative study [28].

It is important to remember that although the process of visualization is a transformation from numerical data to computer graphics, the mapping is most likely to be a one way process and trying to recreate the original dataset from the visualization, could be impossible. It is also important to remember that the visualized data can be potentially inaccurate and is open to human interpretation and perception.

Generally before a dataset can be used, it is necessary to perform some filtering to prepare and clean up the data. Such as interpolating missing data values, removing noise and clamping data to specific ranges.

It is often desirable within a visualization to show a realistic representation of what the data represents, such as modelling prototypes or architectural based data. However it can be more useful to have a simplified representation which could be just as accurate. Other techniques for improving visualization include adding colour and motion.

Adding colour to a visualization could be used to represent a range of values for example representing the temperature of a weather map and thus make the visualized data even more useful to the researcher. Also there are many examples of where realistic colours and image maps have been applied to visualized data to increase realism. By allowing the transparency of different objects to be changed it is possible to give the researcher the feeling of inner structures or skeletons of an object.

Iteration is an important aspect of visualized data that changes over time and actually allows the researcher to manipulate the data using the process of steering, whilst actually seeing the effects in real time.

Figure 2.5 shows two examples of visualizations being done at Bangor University. Figure 2.5(a) shows a model of the Lambeck European Ice Sheet. The model is being used to create a detailed reconstruction of the glacial movements over the last 10 years, with the aim of better understanding about glaciation and the underlying geology [29]. The European Palaeotidal Visualisation Model, shown in Figure 2.5(b) shows an example of where visualization is being used to predict tidal statistics based upon up to date palaeoshorline data [30].

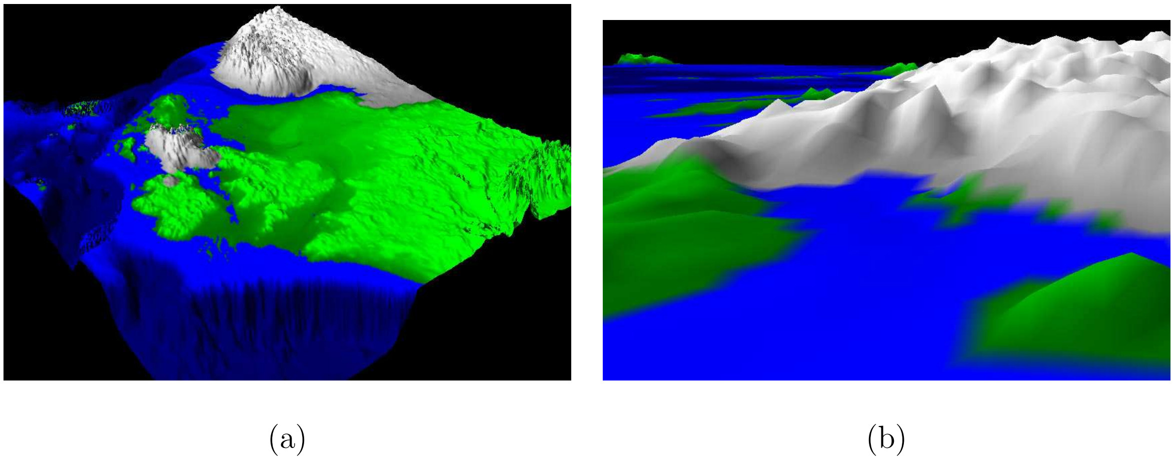


Figure 2.5: Examples of visualization work done at the School of Ocean Sciences at the University of Wales Bangor: (a) The Lambeck European Ice Sheet and Shoreline 3D Visualisation Model and (b) The European Palaeotidal Visualisation Model. Images courtesy of A. Wainwright.

2.4.2 Visualization Model

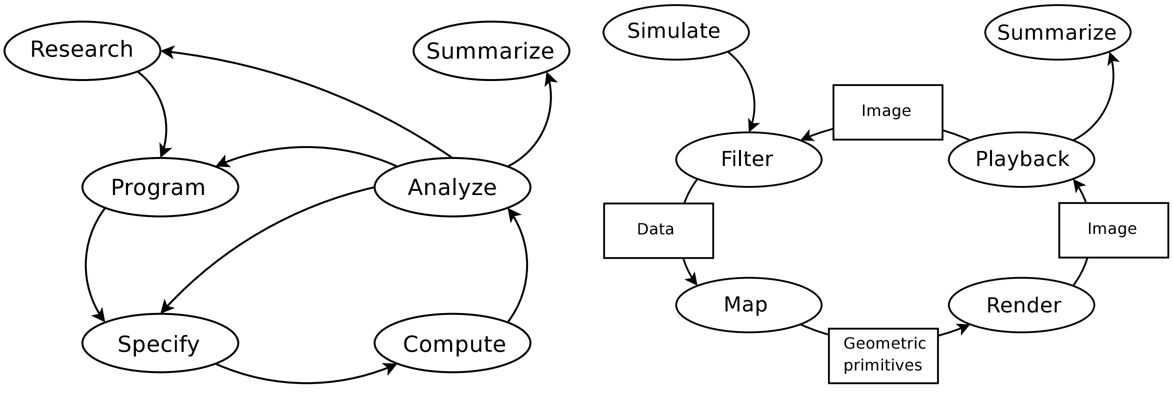
Craig Upson, suggests that in order to ‘deal with the problems of multiple disciplines in the computational sciences effectively, it is useful to begin by developing a coherent picture of the various steps a scientist takes while simulating a natural process using a computational model’ [31].

This simplifies to identifying the similarities between each discipline. Figure 5, demonstrates the process of numerical simulation using a computer program. It shows a circular path which can be repeated many times before a program is finely enough resolved.

The basic strategy to the computational cycle is that after the researcher has performed his initial research, he is ready to program an implementation. This implementation will then produce data which can be analysed. The outcome of this data will decide whether the researcher needs to return to the programming stage, or whether the implementation has succeeded.

The analysis step within the cycle can be broken down into the cycle shown in Figure 2.6(b). This shows that there are several main steps to the analysis of the data [31].

Haber and McNabb [32] defined the linear visualization pipeline that is used for most visualization tasks today. As shown in Figure 2.7, there are four main parts to the pipeline. Firstly the raw data is enhanced to make it suitable for visualization. This includes processes such as calibration, smoothing and interpolation. The derived data is then filtered appropriately and then mapped to a suitable geometry description. The Geometry is then rendered to produce a final image.



(a) (b)

Figure 2.6: Visualization Model: (a) The Computational cycle (b) The Analysis cycle.

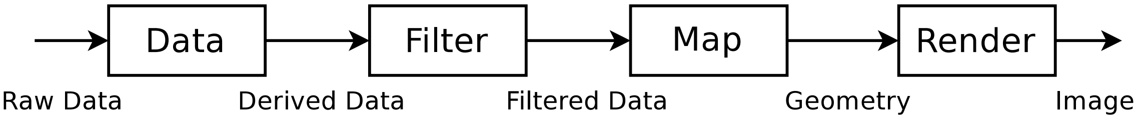


Figure 2.7: Haber and McNabb's visualization model

2.4.3 Human Factors involved in Visualization

Visualization really 'attains its power by captivating the user's attention by inducing a sense of immersion and presence' [6]. This is achieved by using hardware that is able produce stimulations to the main senses of the human body, giving the user a feeling that they are immersed within the virtual environment [33].

There are 5 main senses within the human body each of which are used together to build up the mental image of an environment and it has been found that when more than one of the senses are being stimulated, the user will feel more as though they are actually immersed:

2.4.4 Vision

The Human eye contains a lens which focuses the light on the light-sensitive retina at the back of the eyeball. The light information is then transferred via the optic nerve to the visual cortex [28].

The Human body has Stereoscopic vision. This means that by using two eyes the brain is able to estimate depth by the use of binocular disparity and ultimately gives rise to the sense of 3D vision by providing two different images.

This is the most commonly stimulated sense, which is generally stimulated with the use of a video display which presents the visual information to the user.

Using an ordinary computer display has many limitations, as the user would only see a flat image. To try and extend upon this scientists have developed several ways to send a different image to each eye and thus give the user a 3D perspective of the image.

2.4.5 Sound

The second sense is that of hearing. Human ears allow sound waves to be interpreted by conducting vibrations in the inner ear. The Human ears are also used to tell whether the body is standing upright, or leaning at an angle as well as identifying whether it is stationary or accelerating.

Motion sickness is caused when equilibrium is lost between all of the senses. For example if a person's sight is telling him that he is moving, but his ears say that the body is stationary, then motion sickness can occur. This is an issue that can occur when immersed in a visualization environment, and must be considered.

It is very easy to provide sound within a visualized environment. Most standard computer workstations come with the facilities to produce sound and provided is used to simulate the appropriate environmental sound, it can add a great deal of realism when interacting with a visualized environment.

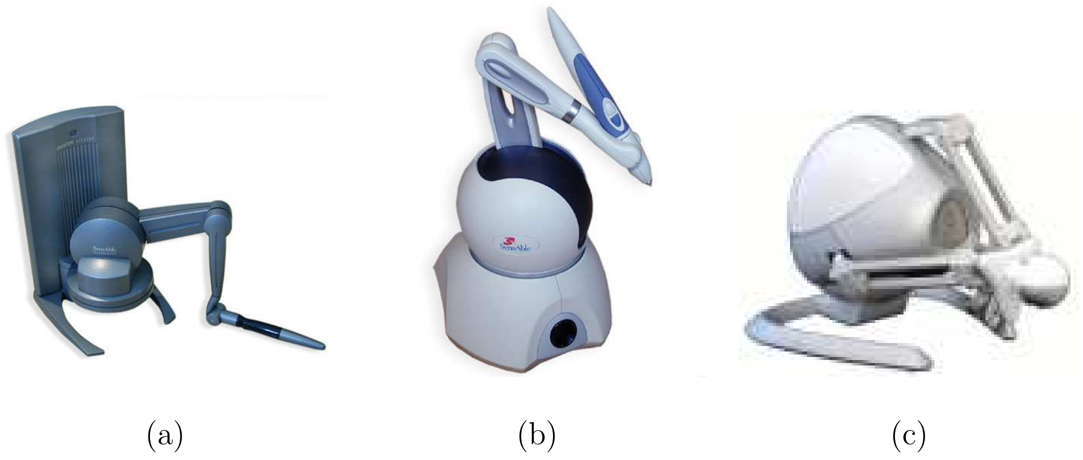
2.4.6 Haptics

Touch is also a very important sense. The human body is made up of a vast quantity of nerves which provide complex feedback from all over the human body. This allows the user to use their hand to actually feel the shape of objects and to identify different attributes such as texture and temperature. This is the most difficult sense to try and stimulate simply because it is very difficult to make it feel like you are actually touching an object without limiting the user's capabilities [34].

The most effective method for giving the user the impression of physically interacting with objects, is to use a Haptic Force Feedback Device such as the Phantom Desktop from SenseGraphics, as shown in Figure 2.8(a). Force Feedback Devices work by allowing the user the ability to freely use a stylus or joystick. However this stylus is fixed to a base and the computer has the ability to provide force back onto the device and depending upon the force used this can produce the feeling of resistance and create the feel of solid objects and different textures.

Lower cost Haptic devices such as the Phantom Omni, as shown in Figure 2.8(b) have been developed in order to make the technology more readily available. Recently a domestic version, the Novint Falcon, has been released to meet the demand of the home gaming industry, shown in Figure 2.8(c).

One limitation that all of these Haptic devices share is that they are restricted to three *Degrees of Freedom* (DOF) as there is only tracking and no force feedback of the stylus itself. Very expensive products such as the Sensable Phantom Premium do provide this functionality but at a cost which makes them unaffordable to many



(a) (b) (c)

Figure 2.8: Examples of Haptic devices: (a) Sensable Phantom Desktop, (b) Sensable Phantom Omni and (c) Novint Falcon.

users.

2.4.7 Input Devices

The user also needs the facility to allow him to interact with the visualization in some way. There are several commercially available solutions to this.

The first is a 3D mouse which is a hand-held device which uses a tracker sensor and a set of buttons. By changing the orientation of this the user could then interact with an environment. For example tilting the mouse forward could be used to control forward movement and speed, and the buttons could be used to allow the user to pick up and use tool objects.

Another example of an input device is an Interactive Glove. This could be worn by the user and transducers sewn into the finger joints can be used to tell the computer the physical characteristics of the fingers when they are bent. This could allow the computer to identify when an object is being picked up [35].

2.5 Problem Solving Environments

The term Grand Challenge was used to describe problems which ‘cannot be solved in a reasonable amount of time with todays computers’ [36]. However it is more commonly used now to describe problems that can be solved cost effectively by the use of parallelism [37]. For the problem to be solved cost effectively, the improvement of the task execution time from that of a single processor must out weight the overall cost of the processors. It must also allow the task to be solved in a reasonable amount of time. Many of the grand challenge applications currently being addressed have been considered to be intractable on a single processor [38] where all solutions could take hundreds of years.

It is suggested that the use of any grand challenge application that may be solved with the use of high performance computer resources will have the potential for broad economic, political, and scientific impact [39] as we would have the capabilities to solve problems that have been impossible before. The exciting potential of these solutions has led to several national grand challenges being proposed giving a competitive edge to the research. HPV is heavily dependant upon data management, distribution and communication and as such has been classified as a grand challenge for many years. However with the HPC resources that are currently available it is now becoming possible.

2.5.1 Problem Solving

Problem Solving Environments (PSE) are ‘computer systems that provide all of the computational facilities necessary to solve a target class of problems’. [40]

Many of the complex problems which fall into the category of Grand Challenge Problems require a PSE which can utilize HPC resources. However this should not over complicate the problem solving process. More specifically, the end user should not need to understand the middleware to use the PSE as this should be taken care of internally [41].

In the field of visualization there are a number of existing PSE's which offer a generic approach to visualization. These are known as Modular Visualization Environments (MVE) and provide user extensible tools [42].

There are a number of commercial MVE's currently available [43]. Although many of them offer similar services, giving the user a library of modules which can be linked together to provide the required function. Some packages us visual programming to give the user a graphics representation of the linked modules clearly showing the path of the data from its raw state to the completed result.

2.5.2 Cactus

Cactus, is an open source PSE, originally designed to provide a 'unified modular and parallel computational framework for physicists and engineers' [44].

The original development of the Cactus code was designed to provide a framework for solving Einstein's Equations. In the early 1900's Albert Einstein first published his theories on relativity and gravity. Part of his claim about gravity suggested the existence of black holes which had such an extreme gravity that nothing could escape, not even light [39].

Up until recently, scientists have just had to accept this theory, being unable to prove it wrong as there was simply no way to be able to solve the equations, within a reasonable amount of time. This has always been regarded as a Grand challenge Problem.

The modular design of Cactus has enabled different teams of experts in the fields of mathematics, physics and computer science to bring together their research and as a result have begun to make progress into this Grand Challenge Problem.

The structure of the Cactus system has been described as a central core (or flesh) which connects to a number of application modules (or thorns). A toolkit is provided with a basic range of thorns. This includes thorns for parallel I/O, data distribution, check pointing and other mathematical functions. Although it is designed to be very easy to implement additional thorn applications such as the applications that were used to help solve Einstein's equations [44].

Cactus is extremely portable and will allow applications, that have been built on standard workstations, to run on clusters and other HPC resources. This is achieved by using a simple API that can be called for features such as I/O operations. Thorns can be written in either C/C++ or F77/F90 which ever is more convenient to the programmer. It also attaches nicely to other technologies such as the Globus Middleware and many advanced visualization tools.

2.5.3 SCIRun

'The SCIRun scientific PSE is a computational steering system that allows the interactive construction, debugging and steering of large-scale scientific computations'.

The SCIRun PSE was initially developed when it was realised that scientists not only want to be able to analyse and interpret results from computationally intensive tasks, but also to be able to steer the simulations as closely as possible to real-time. Therefore a framework was needed that would allow scientists to be able to change the parameters, resolution and representation of data within a simulation and to see what effect it has [45].

In previous PSE's the simulator typically ran within an off-line mode. This meant that the scientist was able to implement a simulation and set the initial parameters. The simulation would then be allowed to run and produce a final result. SCIRun adds the ability to perform interactive steering at each stage of the simulation.

SCIRun also make use of a Data flow System for allowing the programmer to use a visual environment for developing the modular structure in which basic modules can be connected by dragging with the user's mouse. Additional modules can be implemented using C++ and many utility routines are provided to handle the existing SCIRun data structures and basic mathematical computations.

Parallelism is used to make the simulations run as efficient as possible, taking advantage of the resources available within HPC resources. SCIRun does this by allowing different modules to run in parallel, even if this does not explicitly follow the data flow diagram, provided that all of the data for a module to run is available.

2.5.4 Visualization Toolkits

There are also a few specialist visualization toolkits available which all provide very similar features. They are all designed to be usable without specialist programming knowledge, although do provide programming interfaces to allow more experienced user's to extend them further. This generally increases the system overheads as researchers are using a generic solution which has not been optimized for a particular task. The most common tools are listed below:

The Visualization Toolkit (VTK)

The *Visualization Toolkit* [46] is an extensive computer graphics library. It is implemented as a C++ class library and provides interfaces for Tcl/Tk, Java and Python. A large developer base has contributed to VTK making it one of the most complete libraries of visualization libraries including; scalar, vector, tensor, texture, and volumetric methods. Is also provides many advanced modeling techniques such as implicit modelling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation. Although there is a vast range of documentation for VTK, Kitware, Inc. also provide professional support and solutions to non technical users [47].

IRIS Explorer

IRIS explorer [48] provides a visual programming environment allowing the rapid prototype of visualization applications. It was developed by the The Numerical Algorithms Group Ltd (NAG) and as a commercial product it contains many of their world class libraries.

OpenDX

OpenDX [49] is the open source version of IBM's Visualization Data Explorer. Originally released as commercial application IBM have now chosen to release it to the development community to encourage the usage of its Deep Computing range of HPC servers, for visualization tasks. OpenDX was designed for the visualization of scientific and analytical data. It provides a useful Graphical User Interface (GUI) which allows the user's to build complex rendering pipelines.

COVISE

The *Collaborative Visualization and Simulation Environment* (COVISE) [50] provides a distributed software framework which allows the integration of HPC simulations and visualizations within a collaborative environment. It is a modular system with each module forming a separate machine process, allowing each module to be placed arbitrarily within the distributed system. The software provides a Graphical User Interface (GUI) called the MapEditior that allows user's to drag and drop modules to form their visualization pipelines. There are many modules currently available, although the modular structure of COVISE makes it easy to develop new modules for specialist tasks.

OpenGL volumizer

OpenGL Volumizer is an API which has been specifically designed for volume rendering applications. It is a commercial product from SGI and is robust enough to allow researchers to visualize very large data sets.

Other Visualization Libraries

Several other visualization libraries exist offering similar functionality, including TGS Amira [51] and AVS Express [52], both of which are very good at providing visualization solutions, and all provide a graphical interface for designing the data flow between modules [53] [54].

2.6 Components of HPV

Current research into HPV has been looking at specific areas, rather than the complete process as a whole. The following is a list of the most significant development areas:

2.6.1 Computational Steering and Remote Monitoring

Traditionally, intensively computational tasks are non-interactive. This means that once the simulation has been setup, the parameters are set and then the simulation is submitted as a batch, for processing. The simulation will then be allocated the next available processing slot and the simulation will run to completion. The time frame for the simulation will not be real-time, it could be much slower or even faster than real-time. It is also possible that the simulation will get put on hold, whilst other jobs are using the resources [55].

Once the non-interactive simulation has completed, the scientist will receive his results to interpret and analyse. If they then wishe to change any part of the simulation then the parameters will have to be changed and the simulation run again from scratch.

Although this can be useful for some research it can be very time consuming and be an inefficient use of resources. This is where the idea for computational steering has been introduced, to allow the scientist to have some way to interact with the simulation, whilst it is running.

There are two main aspects of computational steering. Firstly to allow the scientist to be able to interact with the simulation there must be some method to allow the parameters to be changed whilst the simulation is running. Secondly in order for the scientist to make informed decisions and to see the result of interaction, there must be some way to monitor the simulation. This is generally done using a visualization of the simulation as it evolves. It is desirable, but not essential, to have have the simulation running as close to real-time as possible to give the scientist the best interaction with the data [56].

The complexity of the visualization can be scaled to meet the hardware resources on which it is processing, to allow the simulation to run in real-time. For example the resolution of the data used could be much higher when running on a HPC resource, than on an ordinary desktop computer [55].

The RealityGrid project [55] is interested in extending the realistic modelling and simulation of complex condensed matter structures using visualization. Computational Steering therefore forms a large part of their research. Their solution lies in the modular design of the applications. In essence, separate applications exist for simulation, visualization and steering each with the ability for communications between components. The simulation component is responsible for generating the data which is sent to the visualization component. The Steering component is able to dynamically connect to either of the other components, which can be both monitored and steered. This allows the simulation to run undisturbed for the majority of the time and only needs to connect whilst the scientist is examining the state of the simulation.

It also provides a library with a variety of features to allow any appropriate applications to make use computational steering. In order for an application to be appropriate it must be able to emit a subset of the simulation values, each value must accept changes and the application must be able to restart from checkpoints [55]. Work is also in progress to see how the RealityGrid can utilize Grid technologies and provide a Steering Grid Service (SGS) to provide a Grid service interface for the component using the Open Grid Services Architecture [57].

2.6.2 Parallel and distributed computation for graphics and visualization

The main limitation within a visualization framework is that of the bottleneck caused by the computational pipeline. This can be solved by combining the visualization technology with HPC resources.

An early project, Visualization in Parallel (Vipar) was set up at Manchester University, to improve upon the issues causing these bottlenecks by providing a series of libraries that would integrate with the current PSE's, such as AVS and IRIS Explorer, and provide improved support for parallelism. Many of the existing parallel solutions where written explicitly for specific tasks and architectures. The Vipar libraries where developed to produce a generic solution which would be portable and scalable [58] [59] [54].

2.6.3 Very large dataset visualization

One of the biggest problems with computationally expensive, visualization tasks is that they can involve very large sets of data. This dataset is usually too big to fit into the physical memory of a computer workstation and has thus rendered many of these tasks as Grand Challenge Problems [60] [61].

At the Georgia Institute of Technology, they have been attempting to provide an interactive fly-through of real life datasets that take up more than 20GB of data. In order to do this it is necessary to organize the data is such a way that it can be transmitted to and from disk quickly enough to support the interactive visualization. The dataset can be stored either locally or across a network. This technique is known as an out-of-core approach to visualization [62].

They found that improved data access and quality of the presentation could be achieved through the use of multi-resolution data that would allow the use of smaller, lower resolution data to be used if the visualization was struggling for time, and the higher resolution data would then load in the background when the disk was not overloaded.

By using techniques such as hierarchical data structures, appropriate memory page sizes and setting priorities to different subsets of data, it was found that they could make vast improvements upon previous efforts.

The research group, are now working towards handling moving data objects, which they believe will integrate easily with the existing system [62].

2.6.4 Collaborative Virtual Environments

Initial work into visualized environments focused on the interaction of one person with the system. However researchers are developing multi-user environments to open up a new range of possibilities. In this way it is possible to have multiple user's sharing and interacting with the same data and visualization whether they are local user's or a great distance apart [63].

Collaborative environments suggest that users are able to connect to the same environment and concurrently edit the same objects. This brings with it several major problems; it is important to handle the distribution of objects and information as well as the delegation of rights and the representation of group structures [64].

There are also several examples of collaborative tasks which would not be possible within a virtual environment. For example an object may in real life require two people to move it, by lifting simultaneously. Whereas in existing virtual environments it is not yet possible to do this.

In order to ensure concurrency it is essential to implement some form of locking.

This means that all of the copies of an object (one at each site), must be locked whilst a user it editing the object. Once the edit is finished, the changes must be forced upon all of the objects after which the lock can then be released. This ensures that all of the objects stay current, and it is not possible to lose any changes [33].

Other techniques for implementing concurrency include: Transaction Mechanisms, Turn-Taking Protocols, Centralized Controllers, Dependency Detection, Reversible Execution and Master Entities [64]. Work at the German National Research Centre for Computer Science has found that the Master Entities technique is most commonly used. This technique implies that only one site (or process) has the rights to make any changes at anyone time. This allows writing and reading of data concurrently, although if a site needs to make changes to the data it must request the write privileges.

2.6.5 Autonomic Visualization

IBM describes a new model for computing called *Autonomic Computing* [65] as a challenge to develop systems that are able to manage themselves according to the goals of the user. When applied to visualization applications this applies to the self-management of the visualization pipeline and to intelligently choose the most appropriate software and hardware for each visualization task. There are four requirements which need to be satisfied in order for a system to be fully Autonomic it must be [66]:

Self-configuring: The system can make intelligent decisions about its own configuration.

Self-healing: The system can recover efficiently from any problems.

Self-optimizing: The system will continuously adapt to find better configurations to meet the user's requirements.

Self-protecting: The system is secure.

IBM describes *Autonomic Computing* as a journey with several stages along the way, as shown in Figure 2.9. The development of *Autonomic Computing* applications can be shown as an evolution through each stage of the deployment model. The current maturity of the e-Viz framework has only achieved the *Adaptive Computing* stage, although its ultimate aim is to eventually be fully *Autonomic*.

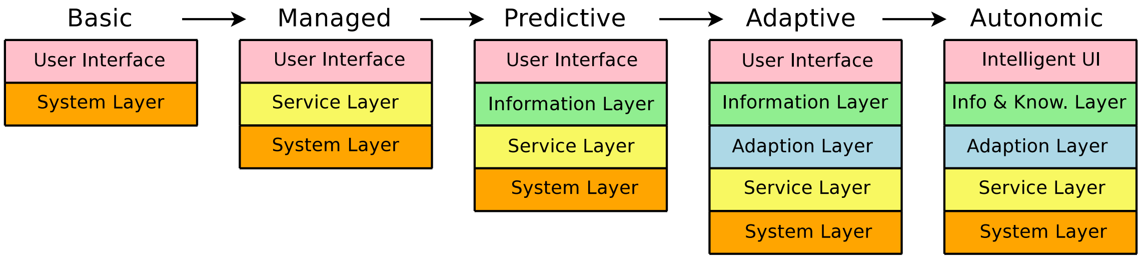


Figure 2.9: *Autonomic Computing* deployment model.

Basic systems provide only a user interface for the visualization task.

Managed systems introduce a service layer, such as the Grid, to manage the security, distribution and available resources.

Predictive systems use an information layer to store performance and quality information about each visualization and use it to make predictions about the best visualization pipeline configuration and which software and hardware resources to use.

Adaptive systems use the information stored by the information layer to self-manage the visualization pipeline and update the system during run-time, using the most appropriate resources available.

Autonomic systems use a full knowledge-base to add a context to the each configuration choice allowing the system to make *intelligent* choices about both pipeline configuration and hardware and software choices.

Most visualization environments rely upon the user having enough knowledge to configure the visualization pipeline and make choices about what software and hardware to use. Roard [67], explores a solution which aims towards autonomic visualization. Rendering strategies to be used are selected by using an agent-based visualization pipeline, which is able to dynamically change strategy during run-time and therefore better adapt to the network conditions. An agent is located on each resource in the system and therefore can add a specific behaviour to each component of the graphics pipeline.

2.7 Summary

This chapter explored the state of the art for AR and looked at many of the supporting technologies, ranging from immersive displays through to the visualization software and hardware required to render the high quality 3D graphics. The chapter looked at how the computational Grid is being used to distribute visualization tasks to remote HPV resources in response to the demand for rendering larger data-sets at higher resolutions, whilst still providing interactive real-time frame rates.

2.7.1 Augmented Reality (AR)

Augmented Reality represents a specific type of immersive environment in which virtual artifacts are superimposed into the real world. In order for the AR environment to be successfully and accepted by the user, three cases must be satisfied:

1. The real and virtual world must be combined with an accurate alignment between the two.
2. The environment must be interactive in and real-time (i.e. more than 15 frames per second).
3. The *Computer Graphics* must be registered in 3D.

There is a vast selection of display technologies that offer some degree of immersion, ranging from a standard desktop monitor with 3D shutter glasses to completely immersive environments such as the CAVE. However in the world of AR it is essential that the display is able to move freely around the real world with the user rather than fixing the user into one place. Therefore the only suitable type of display technology that is realistically available is the HMD, either specifically designed for AR with a see-through design, or using two cameras to capture the user's viewpoint of the real world.

2.7.2 Visualization

A number of visualization toolkits have been developed for to allow the user to create visualization applications. Most of them provide similar functionality and it would be feasible to replicate the same visualization task in each of the different environments, although there is no standard method for specifying each visualization task. Toolkits such as VTK, OpenDX and COVISE all offer free implementations for non-commercial use, although IRIS Explorer, COVISE, OpenGL Volumizer, TGS Amira and AVS Express all require expensive software licenses. This means that often only a specific toolkit is available for use.

IRIS Explorer, OpenDX, and COVISE all offer the user a simple GUI for configuring their visualization pipelines. Whereas VTK and OpenGL Volumizer only provide programming interfaces. VTK does offer TCL/Tk scripting however, which makes it very fast for rapid prototyping.

2.7.3 Autonomic Computing

This review also identified the Autonomic Computing Model from IBM, which states applications much provide four functions to be truly autonomic: Self-configuring, Self-healing, Self-optimizing and Self-protecting. Chapter 7 also discussed whether, or not, there is a truly developed autonomic AR framework which is capable of dynamically distributing its computational and visualization tasks to remote resources.

Chapter 3

Related work

3.1 Introduction

This thesis presents a generic solution for embedding computer graphics artifacts, such as volume rendering, into an AR environment. As shown in Chapter 2, in order to achieve this the following conditions must be true:

1. The real and virtual world must be combined with an accurate alignment between the two.

2. The environment must be interactive and real-time (i.e. more than 15 frames per second is typically required).

3. The *Computer Graphics* must be registered in 3D.

This chapter presents alternative methods for tracking the position of real world objects and how their orientation can be calculated. In order to ensure that the environment is interactive and that the virtual artifacts are rendered in real-time, Web Services and the Grid have also been used to deploy remote HPC resources. Examples of this approach are described below.

3.2 3D Position Tracking

One approach to generating an AR environment is to accurately track a real-world object, and by calculating its 3D position it is possible to superimpose a virtual artifact into the same position in the user's view. Several different technologies exist for tracking real-world objects and mapping their 3D position into the computer software:

3.2.1 Mechanical Tracking

Mechanical tracking, allows the position of a real world object to be calculated very precisely by the use of a physical connection between the tracked object and a fixed reference point. Usually a mechanical arm with multiple points of articulation is used, allowing free movement of the tracked object by the operator. At each point of movement, a potentiometer accurately records the angle of each link of the arm and this information is used to calculate the 3D position of the object.

Bajura, Fuchs and Ohbuchi [68] demonstrated how mechanical tracking can be useful for tools within a restricted area, where the limitation of a mechanical arm does not restrict the operator. They developed a training tool for ultrasound guidance for interventions, such as lesion biopsy and cyst aspiration. Using mechanical tracking of the ultrasound probe allowed for the position and orientation that the operator was holding the probe to be accurately calculated. This information allowed them to generate a simulated ultrasound view of a virtual patient realistically based on the position that the operator was holding the ultrasound probe.

Haptics devices, such as the Sensable Phantom Desktop and Omni (as described in Chapter 2) are also examples of general purpose Mechanical Tracking devices. They provide the operator with a generic stylus that can be easily replaced with other tools. Several API's are available for developing software applications that use these Mechanical tracking devices and provide the programmer with the calculated 3D position of the tool, such as the OpenHaptics toolkit [69], Chai3D [70], H3D [71] and SOFA [72]. A comprehensive review of these software tools can be found in [73]. In addition to measuring the position of the tool these devices also provide a force feedback, which allows the user to interact with virtual objects. Work has also been done to use the force feedback to produce an *anti-gravity* effect [74]. By compensating for the weight of the arm it is possible to create the illusion that the tool is freely movable and not tethered to a fixed point.

3.2.2 Magnetic Tracking

Electromagnetic Tracking systems, use multiple coiled wires which have been arranged in perpendicular orientation to each other and embedded in the object which is to be tracked. An electric current is cycled through each of the coils generating a small electromagnetic field. Sensors within the tracking system measure the magnetic field generated by each of the wires, and use this information to determine the position and orientation of the sending unit. Electromagnetic tracking devices are generally regarded to be very accurate, providing tracking of multiple objects which are unrestricted in their movement within the tracking area. They are however susceptible to interference from both electronic devices, which can create their own electromagnetic fields, and metal objects which can cause disturbances to the magnetic fields [75].

The coils which are used to generate the electromagnetic fields can be driven by either Direct Current (DC) or Alternating Current (AC). This results in fields which are either constant (DC) or continually changing (AC). The main advantage for using a constant (DC) magnetic field is that no eddy currents are generated in the surrounding metallic objects. However DC fields can be affected by the environmental magnetic field of the Earth and therefore must be compensated for. With an AC magnetic tracking system, eddy currents are generated, but are frequency dependent and these dependencies are useful in calibrating the system to reduce the distortion effects on accuracy [76].

One example of a commercial application, developed by Innovative Sports Training (IST), is a Biomechanics Analysis tool as shown in Figure 3.1. It uses the Polhemus Liberty magnetic tracker, to acquire key positions from a player during several sporting movements. This information is then analyzed by mapping it to a virtual skeleton within a visualization environment. By performing this analysis the aim is to improve upon the players efficiency by understanding better how the body moves [77].

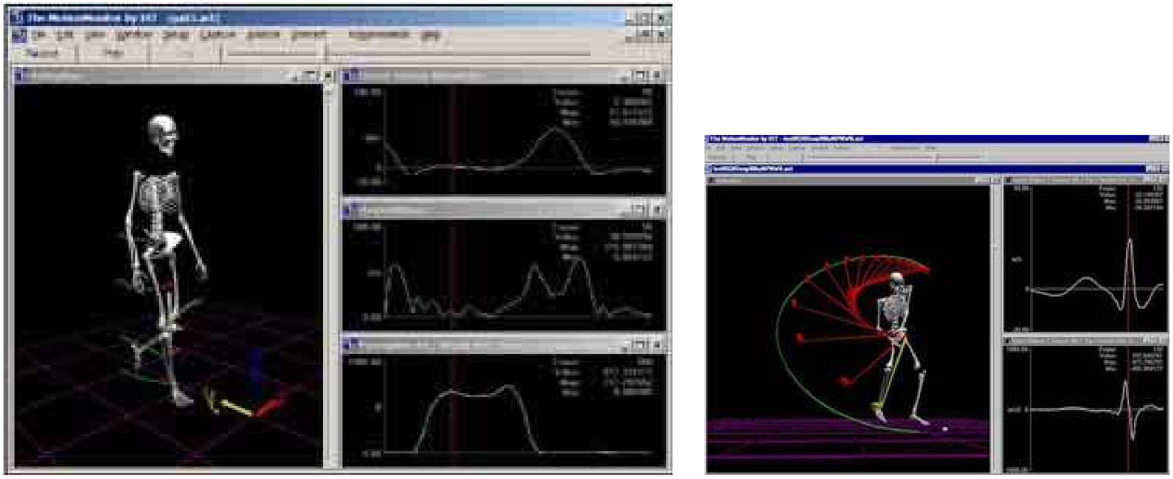


Figure 3.1: A Biometric Analysis tool from Innovative Sports Training (IST).

Several other proprietary Electromagnetic tracking systems are available, costing between $2,000 and $10,000. Popular examples include the Polhemus Fasttrak [78], NDI Aurorta [79] and the Ascension Flock of Birds [80].

Magnetic Tracking is not suitable for medical applications, where the electromagnetic radiation can cause interference with other medical facilities.

3.2.3 Inertial Tracking

Inertial tracking provides a method for being able to track the actual movement of a user or object, relative to its previous position. The movement is calculated using the data generated by a combination of gyroscopes and accelerometers which are located on each of the three perpendicular bodies of the tracked object. Although several research groups have attempted to use inertial tracking for Augmented reality applications, very few have found that using inertial tracking alone provides enough information. Projects such as [81], showed that when combined with other tracking technologies such as optical tracking, the Inertial movement information of the user's viewpoint can increase the accuracy of the tracking.

Several commercial Inertial Tracking devices are available, with the most popular being the wireless Intersense InertiaCube [82] which costs around $2,000, as shown in figure 3.2.

Biomedicom, a commercial enterprise based in Isreal, have been using Inertial Tracking as part of their 3D ultrasound work. By placing the Inertial Tracker inside a ultrasound probe, it was possible to calculate the probe's position and therefore produce a 3D ultrasound image whilst using a standard 2D scanner [83].

3.2.4 Acoustic Tracking

Acoustic Tracking devices use ultrasonic sound waves to determine the position of a target object. These sound waves must have a frequency about 20kHz, which is too high to be audible by a human ear. There are two methods that can be used:

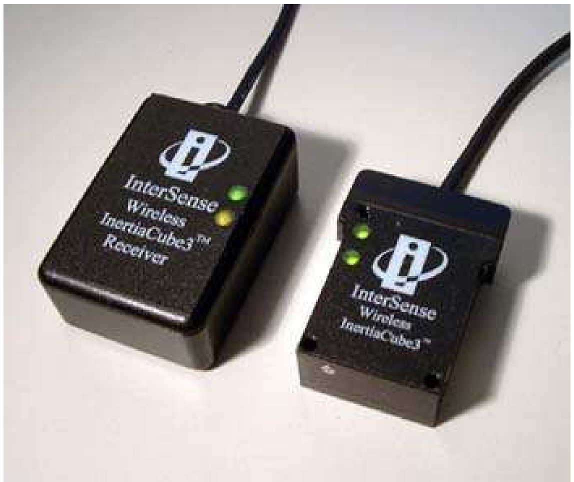


Figure 3.2: The Wireless InertiaCube from Intersense.

Time-of-flight and phase coherence.

Time-of-flight tracking, measures the amount of time taken from sound emitted from a transmitter to reach the target. By using multiple transmitters sequentially transmitting sound, it is possible to calculate the distance from the target to each of the transmitters which can be used to find the precise position of the target. This method generally suffers from a low update rate caused by both the relatively slow speed of sound in air, and the environmental impact on the speed of sound, such as temperature and humidity. Phase Coherence improves upon this by measuring the difference in phase between the sound waves at the transmitter and at the target. As long as the distance between updates is less than a complete wavelength, it is possible to calculate the distance based upon the current position in the sound wave.

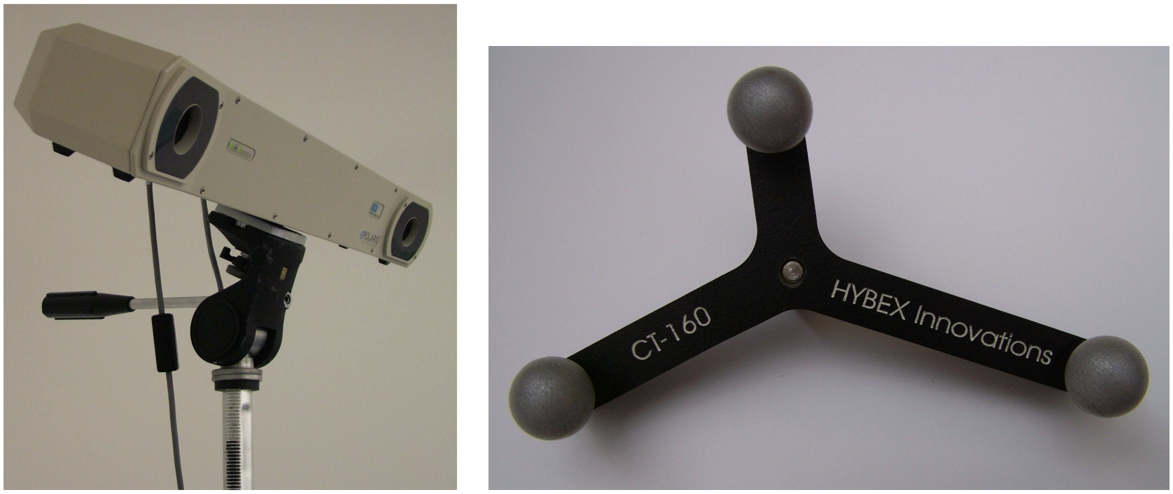
Alusi et al. used acoustic tracking to calculate the position of a handheld ultrasound probe. This information allowed them to develop a novel AR application for skull base surgery [84].

3.2.5 Optical Tracking

Optical Trackers can be split in to two categories: Beacon tracking and Computer Vision tracking.

Beacon Tracking

Beacon trackers rely upon the use of a set of optical beacons. These can be either active or passive, depending upon whether they transmit light or simply reflect it.



(a) (b)

Figure 3.3: (a) The Polaris Optical Tracking system and (b) a generic marker which will be used to track the electromagnetic coil.

The Polaris Optical Tracking system from NDI [85], as shown in Figure 3.3, is an example of a commercially available passive tracking system. It is designed to track wireless tools within a limited optical range. Tools are defined by a unique pattern of markers fixed in position on a rigid body. The tracking unit emits a constant infra-red (IR) light that is reflected by the markers on each tool, back to a position sensor - see Figure 3.4. The Polaris system is then able to use triangulation to calculate both the position and orientation of each tool within the working area.

The Polaris system provides the programmer with 6 Degrees of Freedom (DOF) for each tracked tool: the translation in all three perpendicular axes (X, Y and Z) combined with rotation about the three perpendicular axes (yaw, pitch, roll). The TMS application described in Chapter 4 uses this Polaris Optical Tracking System.

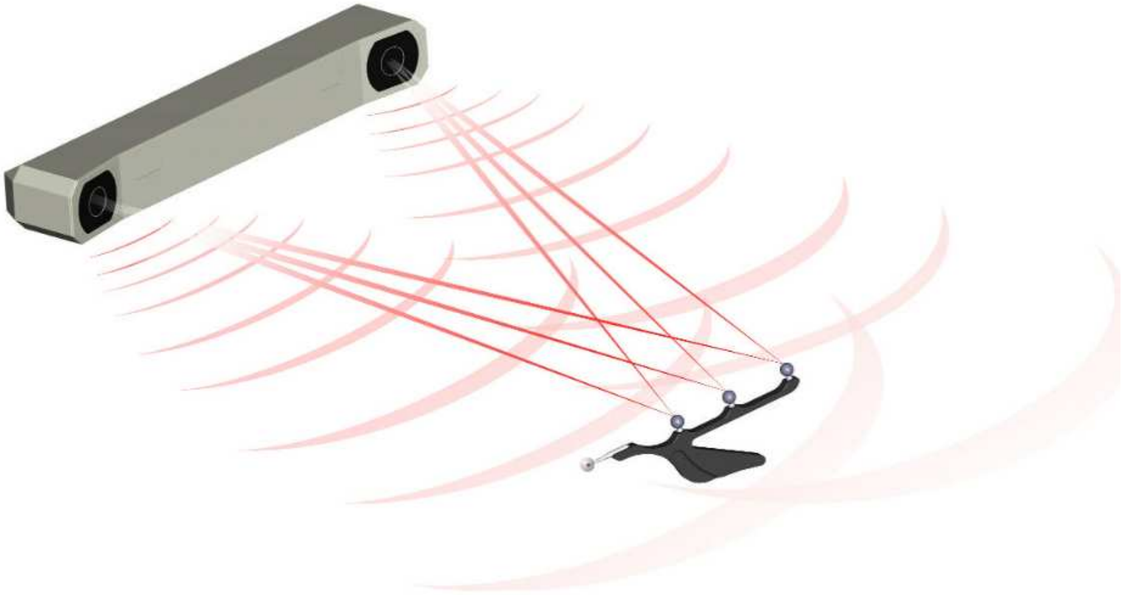


Figure 3.4: The Polaris system emits IR light which reflects back from the markers on each tool.

In order to improve accuracy it is possible to use active tracking, where infrared LED's are positioned within the view of a camera. By pulsing the LED's individually it is possible for the tracking system to identify rapidly which part of the object being tracked is in view. At the University of North Carolina, a research group have developed a technique for tracking the head of the user by mounting three cameras on a HMD, each pointed towards the ceiling. The ceiling is then covered with 1000 uniformly distributed Infrared LED's which are each cycled during tracking, and by understanding exactly where each camera is pointing it is possible to accurately calculate the position and orientation of the HMD [86].

Computer Vision Tracking

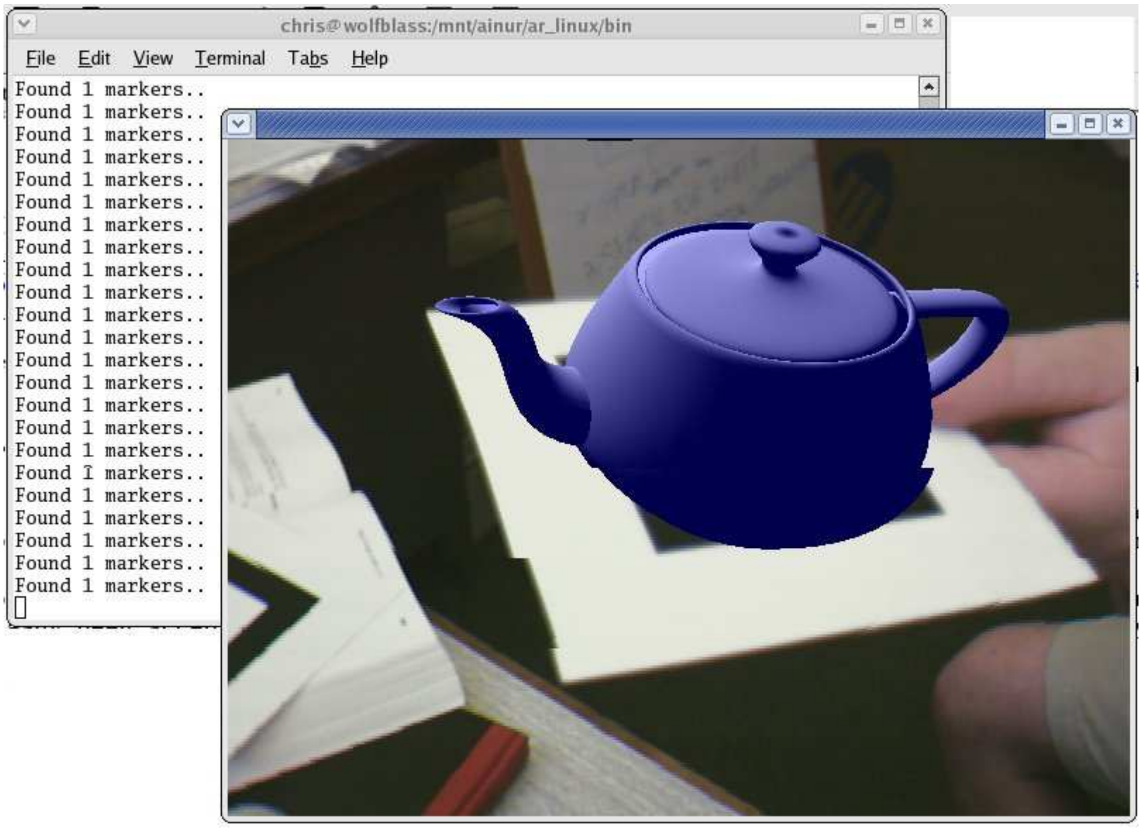
Computer Vision techniques allow objects to be tracked by identifying them within a camera viewpoint. Several Toolkits exist that allow objects to be tracked using only a cheap, easily available web-cam. The most common of these include:

OpenCV - The *Open Source Computer Vision Library* (OpenCV) from Intel, although not strictly a library for developing AR applications, it does provide many of the libraries required for processing real-time camera images. It provides libraries to perform computer vision tasks such as: Object Identification, Segmentation and Recognition, Face Recognition, Gesture Recognition, Motion Tracking and Understanding, Structure From Motion (SFM) and Mobile Robotics [87].

ARToolkit - The Human Interface Technology Lab (HIT Lab) as Washington university has developed a technique allowing 2D Markers, or 'Tiles' to be used to as a reference between the virtual object and the real world. Using a head mounted camera the computer can identify the uniquely labelled Tiles and superimpose other graphics onto each Tile [88]. The HIT Lab have also produced a software library for building AR applications called ARToolKit which is freely available [89].

osgART - The osgART *Software Development Kit* (SDK) [90] was also developed at the HIT Lab as an extension to the ARToolkit. It combines the ARToolkit with *OpenSceneGraph* (OSG) [91], a high performance graphics toolkit to allow high quality rendering and support for many standard file types.

DART - The *Designer's AR Toolkit* (DART) [92] is aimed more at artists and non-technical designers, by providing a graphics development environment for devel-


chris@wolfblass:/mnt/ainur/ar_linux/bin
File Edit View Terminal Tabs Help

```
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
Found 1 markers..
```

Figure 3.5: A virtual teapot is aligned with the real world using a marker detected by the ARToolkit.

oping AR environments. The toolkit uses specified markers which can be attached to landmarks or objects. DART is provided as a series of extensions to Macromedia Director [93], allowing sophisticated multimedia content to be associated with the markers, including 3D graphics, sound and video, in an environment which is already familiar to most multimedia designers.

MRT - The *Mixed Reality Toolkit* (MRT) [94] was developed at *University College London* (UCL) and is the first toolkit to move away from the requirement to use specific markers. Instead it uses a primitive modeling technique to identify simple geometric shapes within a video stream. This makes it very effective for tracking simple objects such as cubes or tables, but unsuitable for identifying more complex or generic objects. It also relies mainly on using a fixed camera, which makes it difficult to use when the operator wants to move freely within an AR environment.

BazAR - The BazAR framework [95], is a computer vision based library based upon feature popint detection and matching. By using textured plane surfaces as markers it allows markers to be easily calibrated by capturing a good quality view of each marker being used. BazAR uses a Scale-Invariant Feature Transform (SIFT) algorithm to detect and describe feature points within the video stream [96]. Dementhon's Pose from Orthography and Scaling (POSIT) algorithm is used to calculate the pose of the tracked object based on the set of extracted feature points.

InstantReality - The InstantReality [97] is a framework suited to both VR and AR. It brings together a number of different techniques to provide both marker and marker-less based environments, and is compatible with a number of different immersive display technologies. The system is compatible with a number of industry standards, such as VRML, X3D and OSG.

3.2.6 Tracking Summary

There are advantages and disadvantages to each of the tracking techniques explored here. Mechanical, Magnetic, Acoustic and Optical Beacon tracking methods all impose restrictions upon the work space area that can be used to track the object. Inertial tracking however places no such restriction on the work space because it only provides the movement of the object, rather than calculating a position for the object.

Commercial products exist for each type of tracking device, although they tend to be very expensive. The computer vision tracking techniques are possible with a very cheap and easily available web-cam making it the only option for a cheap AR application. Mechanical tracking is the only technique which does not suffer from environmental interference, such as magnetic interference in magnetic tracking, or changing light levels with optical tracking.

Each of the Optical tracking techniques rely upon the use of markers to be added to the real world scene, which can be intrusive. They also have a much higher computational requirement, as it is much harder to search for the object within the camera's view.

Several software toolkits exist to help with the development of AR applications using optical tracking. Most of these toolkits, such as ARToolkit, osgART and DART all provide functions for aligning virtual objects with specific markers which have been defined in the application. MRT is the first toolkit to allow the user to track objects which are not specifically aligned with marker, however the toolkit is limited to only identifying strict geometric shapes such as cubes and tables. BazAR was the first AR framework to use extracted feature points for tracking the real world. It is however limited to only using plane surfaces reliably, rather than tracking 3D objects in the real world.

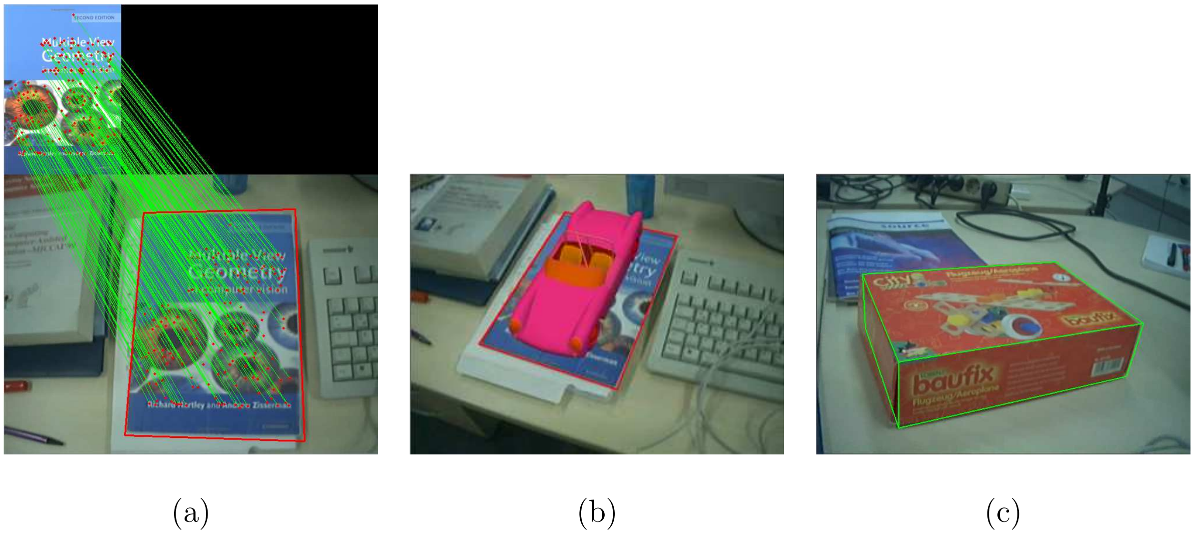
Several other toolkits exist such as InstantReality which brings together several of the technologies described above and allows the user to combine them in a single environment. The OpenCV library provides functions for image processing and real time processing of camera images, although it is not specifically designed for AR applications.

The first software tools were all deigned around simple OpenGL applications, although the more recent ones do provide support for other tehchnologies, such as VRML, X3D anf OSG. None of them currently provide facilities for integration with

HPV resources, without modification.

3.3 Pose Estimation

When using a computer vision based tracking approach, it is necessary to reconstruct the 3D position and orientation of the object that we are tracking. There are several methods for doing this, as illustrated in Figure 3.6.



(a) (b) (c)

Figure 3.6: Three common methods for Pose Estimation: (a) Point-based, (b) Contour-based and (c) Texture-based.

Point-based tracking is based upon the detection and registration of feature points between those extracted from an image and a set of reference points. Contour-based tracking, detects the object boundary lines, or contours. Texture-based tracking uses techniques which register the whole 3D textured surface of the image to the virtual model. Yilmaz, has completed a full review of these techniques [98].

3.4 Feature point Extraction

Many methods attempt to extract feature points from an image and although there is no set definition of what a feature is, it is generally accepted that a feature is simply a point that can be repeatedly identified from an image [99]. Since the 1970s many feature point detectors have been proposed and as a *good* detector is yet to be found, there is still work today to improve their accuracy and efficiency. Feature points can define either a single pixel point within the image or a larger region of interest. There are three main categories of feature points that can be extracted from our image data, as shown in in Figure 3.7.

Edge Points Edges are defined where there are boundaries between two regions of the image. Generally speaking this defines points within an image that have a high gradient magnitude.

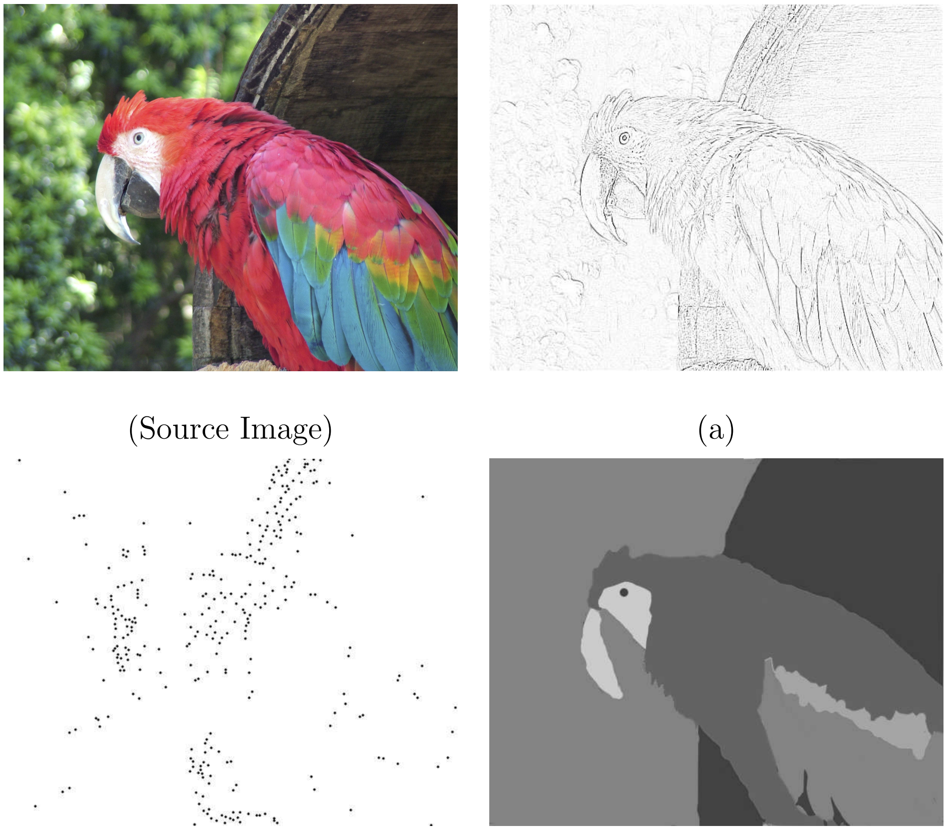
Corner Points Corner detection operator's build on edge detection ideas to define a corner as a single pixel point at each edge intersection.

Area Points Areas (also referred to as Blobs) provide more general descriptions of structures and areas within the image.

There are several methods for detecting feature points in images that all follow these steps, as illustrated in Figure 3.8:

1. The source image data is captured and desaturated into a two-dimensional array of grey-scale values.

2. For each point in the image data a *Cornerness* value is calculated by the operator, and relates to how likely it is believed that that point is a corner.



(Source Image)

(a)

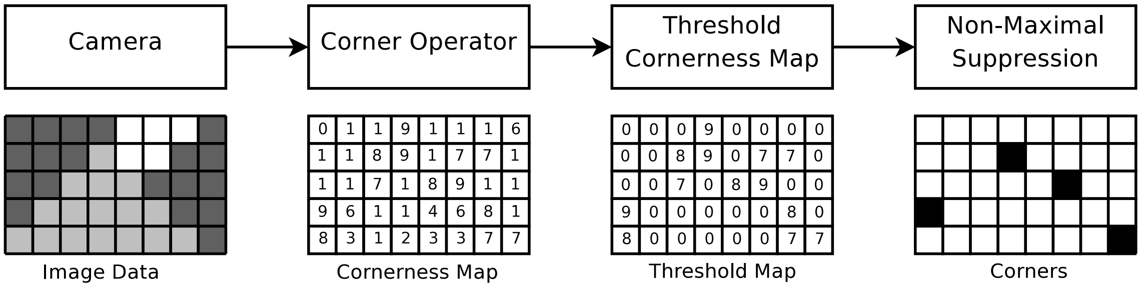
(b)

(c)

Figure 3.7: The three main categories of feature points: (a) Edge points, (b) Corner points and (c) Area points.

3. A threshold value is used to disregard any points that are identified but are not strong enough to be true corners. The *Cornerness* value of these points is then typically set to zero.

4. Non-maximal suppression sets the *Cornerness* value for each point to zero if its *Cornerness* value is not larger than the *Cornerness* measure of all points within a certain distance. This ensures that we only find maximum points and so we can then assume that all non-zero points are corners.



Cornerness Map

0	1	1	9	1	1	1	6
1	1	8	9	1	7	7	1
1	1	7	1	8	9	1	1
9	6	1	1	4	6	8	1
8	3	1	2	3	3	7	7

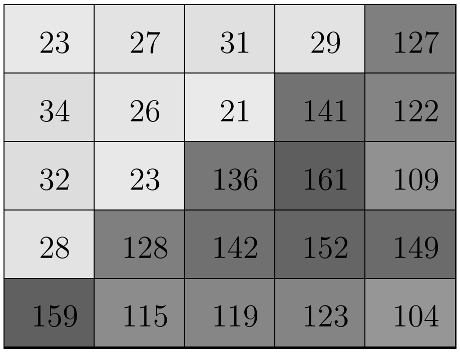
Threshold Map

0	0	0	9	0	0	0	0
0	0	8	9	0	7	7	0
0	0	7	0	8	9	0	0
9	0	0	0	0	0	8	0
8	0	0	0	0	0	7	7

Figure 3.8: The steps involved in feature detection.

3.4.1 Convolution

Kitchen and Rosenfield [100] used differential geometry operator's to detect edges within an image, identifying corners at the intersection of two edges. An edge can be described as an linear intensity change over a series of pixels. In order to efficiently find these intensity changes, first and second order derivatives of the image data are used to generate a gradient map representing the intensity changes within the image. For example it is possible to instantly identify the edge in the following segment of a grey-scale image:



23	27	31	29	127
34	26	21	141	122
32	23	136	161	109
28	128	142	152	149
159	115	119	123	104

Figure 3.9: A subset of a grey-scale image showing an obvious edge.

The first order derivative of an image is generally used to calculate an intensity gradient for the data, which can be searched for peaks and troughs that could represent an edge. If $\mathbb{I}(x, y)$ represents the grey-scale value for pixel x, y then the first derivative $\mathbb{I}_1(x, y)$ can be calculated as:

$$\mathbb{I}_1(x, y) = -\tfrac{1}{2} \cdot \mathbb{I}(x-1, y) + \tfrac{1}{2} \cdot \mathbb{I}(x+1, y) - \tfrac{1}{2} \cdot \mathbb{I}(x, y-1) + \tfrac{1}{2} \cdot \mathbb{I}(x, y+1)$$

This is the equivalent of applying the following convolution mask to the initial image data:

0	$-\frac{1}{2}$	0
$-\frac{1}{2}$	0	$+\frac{1}{2}$
0	$+\frac{1}{2}$	0

By computing the second derivative we can take this one step further and calculate the rate of change for the gradient intensities. Peaks and troughs found in the second derivative can therefore be taken to represent lines in the image, where there is a rapid change in gradient on both sides of the line. If $\mathbb{I}_1(x, y)$ represents the first derivative value for pixel x, y then the second derivative $\mathbb{I}_2(x, y)$ can be calculated as:

$$\mathbb{I}_2(x, y) =$$
$$1 \cdot \mathbb{I}_1(x-1, y) + 1 \cdot \mathbb{I}_1(x+1, y) + 1 \cdot \mathbb{I}_1(x, y-1) + 1 \cdot \mathbb{I}_1(x, y+1) - 2 \cdot \mathbb{I}_1(x, y)$$

This is the equivalent of applying the following convolution mask to the first derivative data:

0	$+1$	0
$+1$	-2	$+1$
0	$+1$	0

Depending upon how clean or noisy the source image data is, calculating the first and second derivative will still produce a number of incorrectly identified points.

Therefore a threshold must be used to remove irrelevant features. A common solution to problem of finding where to set the threshold is to use hysteresis [101]. An initial point on an edge is located using a simple threshold. The algorithm then passes through each point following the edge. Providing the image contains continuous edges then it ensures only lines are located and no other erroneous points are found.

3.4.2 Auto-correlation

Moravec proposed a simple algorithm in 1977 as part of his work on machine vision enabled robotics [102]. He suggested that a point could be identified as a feature point if there was a significant intensity variation in each direction from that point. Although this algorithm provides basic feature detection without being too computationally intensive, it is not repeatable as the points it finds are only repeatable when the edges are at 45^o or 90^o to the point being evaluated. Harris [103] improved this method but at a significant cost to the computational requirements.

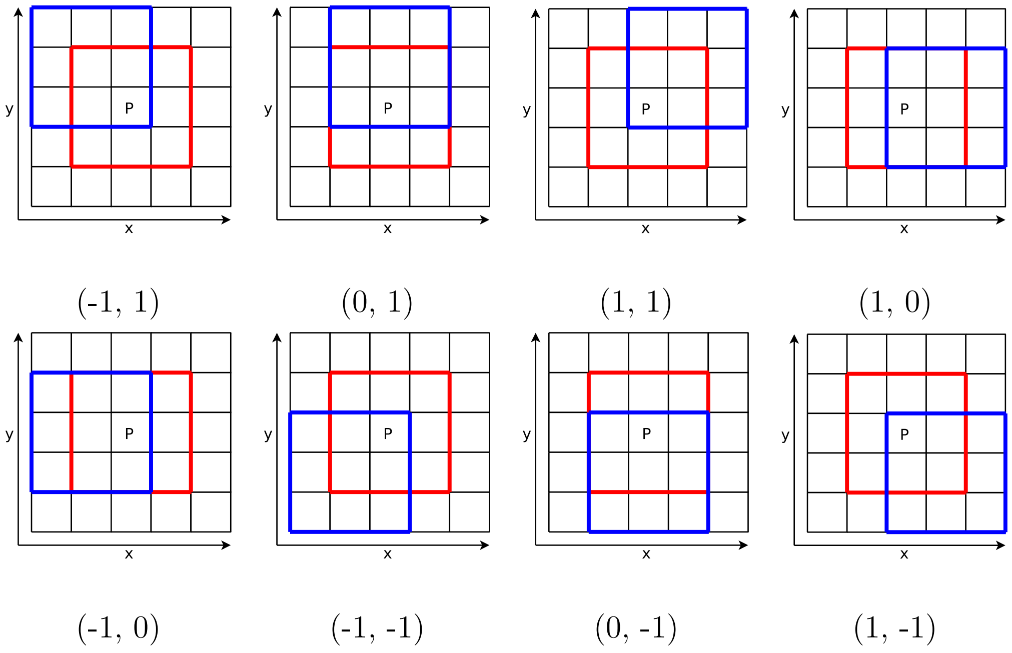
Year	*Corner Detector*
1977	Moravec
1988	Harris / Plessey
1997	SUSAN
1998	Trajkovic and Hedley
1999	Zhenq and Wang

Figure 3.10: Timeline of Corner Detectors.

More recently a significant number of approaches for detecting corners have appeared, as shown in Figure 3.10. The most common of these are summarized below:

Moravec operator

Moravec calculated an intensity variation for each pixel within the image. This was done by placing a fixed size square over the image data centered on the pixel that we are calculating the intensity variation for. This square is then shifted in each of the eight possible directions, as shown in Figure 3.11 and the intensity variation for each shift is then calculated by the summation of the square value of the difference in value between the each of the cells and it original position.



(-1, 1) (0, 1) (1, 1) (1, 0)

(-1, 0) (-1, -1) (0, -1) (1, -1)

Figure 3.11: The eight positions shifts required to calculate the intensity variation of the centre point.

Therefore the intensity variation for pixel $P_{(x,y)}$ can be calculated as:

$$P_{(x,y)} = \sum_{i=-1}^{1} \sum_{j=-1}^{1} (P_{(x+i,y+j)} - S_{(x+i+u,y+j+v)})^2$$

For each (u, v) in $((-1, 1), (0, 1), (1, 1), (1, 0), (-1, 0), (-1, -1), (0, -1), (1, -1))$ where (u, v) represent each of the eight shifted positions. P represents the (3x3) box around the specific pixel (x, y), S represents the shifted box and (i, j) represent each cell of the (3x3) box. The minimum intensity value found from each of the eight shifts is stored as the Cornerness value. A threshold value is then used to remove any points that are below a specific value. Finally non-maximal suppression [104] is used to find all maximum points within its locality. Figure 3.12 provides an example of the Moravec operator in use.

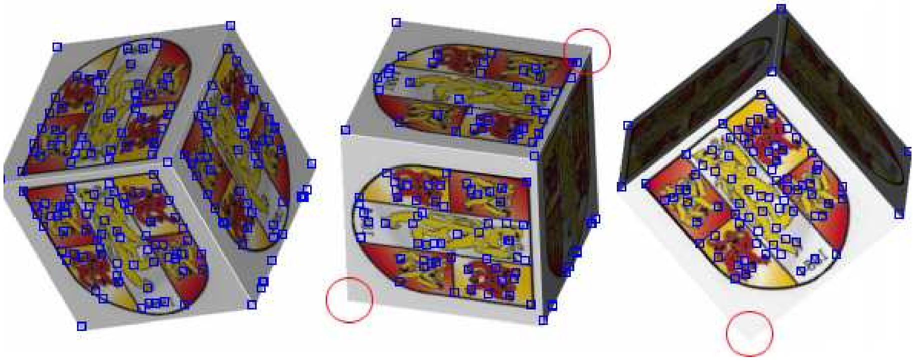


Figure 3.12: The Moravec Operator used to extract corner points from a spinning cube, with some obvious points missed and circled in red.

The Moravec operator provides the basic procedure that we use for our feature point detection, however, due to its nature of using the eight shift position pattern it only becomes reliable when the edges in view are at 45^o or 90^o to the point being evaluated. We need a detector that is able to detect repeatable feature points even when the operator is making subtle viewpoint changes, which will never occur in 45^o steps.

Harris / Plessey operator

Harris and Stevens extended the Moravec detector by addressing the issue of the directional dependence and developed a new method for how the auto-correlation is calculated allowing the variation intensity to be obtained over different orientations. The first order derivative is used to find an initial estimation of the gradient of the image. The Moravec operator is then used to measure the intensity variation within the first order derivative. Figure 3.13 demonstrates the improved performance obtained with the Harris Operator.

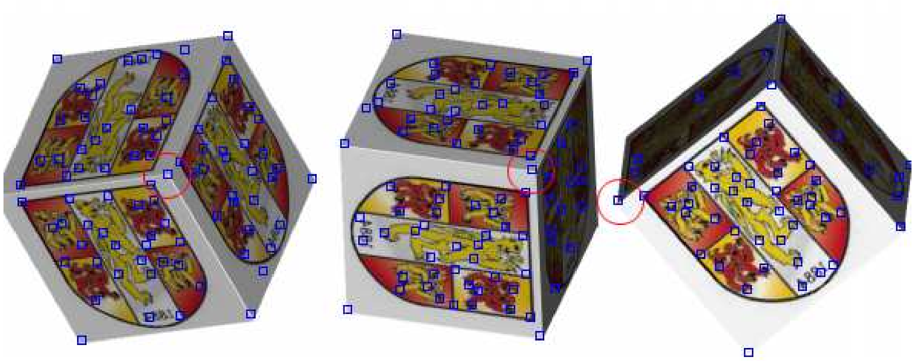


Figure 3.13: The Harris Operator used to extract corner points from a spinning cube. An example of a point which has been accurately repeated is circled in red.

The Harris operator is more computationally expensive than the Morovac operator, although it does yield more accurate results.

SUSAN

The Smallest Univalue Segment Assimilating Nucleus (SUSAN) algorithm makes the assumption that within a small circular region pixels belonging to a specific object will have relatively uniform brightness. SUSAN uses a circular mask which is placed over the pixel to be tested, which is known as the nucleus. The number of pixels with a similar brightness to the nucleus are calculated. Corners are then detected by applying the mask to each pixel within the image to produce an Univalue Segment Assimilating Nucleus (USAN) map. Each local minima within the USAN map is then marked as a corner.

SUSAN is relatively robust from noise, quick to compute but has only a very average repeatability rate [105].

Trajkovic and Hedley

Trajkovic and Hedley improved upon the SUSAN algorithm, by considering the brightness along all lines passing through each nucleus, assuming that at each corner the variation in brightness will be high for each line. Although the repeatability rate of this algorithm is no better than SUSAN, it is one of the fastest algorithms available [106].

Zhenq and Wang

Zhenq and Wang attepted to address the computational complexity of the Harris Operator. By identifying the key aspects responsible for corner detection they were able to reduce the computational complexity at a cost of a slightly degraded corner detection performance [107].

3.4.3 Feature Point Extraction Summary

The Harris operator is generally accepted to be the most robust corner detection operators developed, however it is very computationally intensive. In order to reduce its computational requirements several researchers have tried to develop more efficient algorithms, although they have all come at a cost to reliability and repeatability.

3.5 Web Services and the Grid

This thesis explores an hypothesis that it is possible to use Grid computing for AR applications. Computer Vision can be applied to AR tracking (as described above) but typically this process is computationally heavy. The remainder of this chapter explores the state of the art of Web Services and the Grid. The intention is to determine whether or not this approach is suitable for supporting the computation required in AR.

3.5.1 Web Services

Web services provide a standard means of communication over a heterogeneous network of computers, providing interoperability between a variety of hardware architectures and software packages. The World Wide Web Consortium (W3C) provides a common definition of the Web Service Architecture (WSA), providing a conceptual model of how Web Services should be implemented, as shown in Figure 3.14 [108].

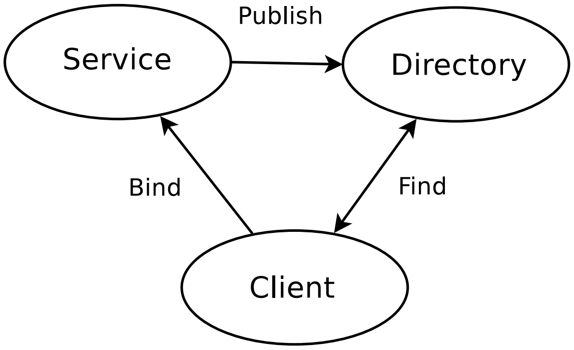


Figure 3.14: The Web Service Concept.

In this basic concept, three steps occur:

1. The Service advertises itself in the Directory
2. The Client user the Directory to find an appropriate Service.
3. The Client starts interaction with the service.

Web services use a service orientated Architecture (SOA) to provide interaction between agents, which can be either software or human, by passing messages between them using XML. This is implemented using three main technologies:

SOAP

Although initially SOAP used to be an acronym for Simple Object Access Protocol, the technology has now evolved and is now no longer regarded as simple and no longer uses objects. SOAP is a simple framework for exchanging XML based messages between agents. SOAP generally uses the HTTP protocol for transporting information, although any valid Internet application layer protocol could be used, such as HTTPS which could make use of the underlying encrypted transport protocol. It is both platform and language independent, although many specific implementations exist, such as gSOAP [109], SOAP Toolkit [110] and Apache Axis [111].

Web Service Description Language (WSDL)

WSDL is an XML based language used for describing network services, including details of the services capabilities and locations. It defines each service as a network endpoint capable to exchange messages. It is designed to communicate either using SOAP, HTTP or MIME [112].

Universal Description, Discovery, and Integration (UDDI)

UDDI provides a registry mechanism for clients and servers to establish contact, using SOAP for communication. A popular implementation is Apache jUDDI [113].

3.5.2 The Grid

Ian Foster, is one of the central characters in the development of the Grid. He describes, the Grid, as a distributed computing infrastructure for 'coordinated resource sharing and problem solving in dynamic, multi-institutional organizations' [114] [41].

The concept of the Grid is still an emerging, new technology. This is because of two main reasons; firstly potential user's are still struggling to see how applications will be able to benefit from Grids and secondly because the available Grid software is still hard to use and not application specific [115].

The general purpose behind the Grid is to provide a middleware infrastructure for allowing the sharing of resources that are not necessarily local to the user's in geographic terms. This has two major advantages. Firstly not every organization will need to have their own High Performance Computing resources and secondly by allowing extra user's to use the resources can prevent the resource being wasted by filling up any free processing time. Also if organizations get together to buy and share resources it generally means that better and more powerful resources become available.

However when implementing Grid technologies, the only aspect that the whole Grid community has agreed upon is that the IPv4/IPv6 is the best current choice for the underlying protocol. There is still a great deal of experimentation going on to decide the best way to provide middleware services and as such there are as of yet no actual standards, even though it is generally agreed upon what actual services are required [116].

The Open Grid Services Architecture (OGSA) has been set out to try and map a new path for development for both the grid and web services. The OGSA has been set in place to define standard communication protocols and formats, to ensure that all of the current research work and developments will be compatible in the future. The OGSA is part of the Global Grid Forum (GGF) and as such produces a detailed set of documents detailing these standards which are currently being instantiated in the Open Grid Services Infrastructure (OGSI) [57].

The Open Grid Services Infrastructure (OGSI) represents the Web Services Description Language (WSDL) specification which defines the standard Grid Service interfaces, behaviours and schema. This is kept in line with the OGSA vision. The OGSI Specification is maintained by the Global Grid Forum (GGF) where it has been implemented on a number of platforms [117].

3.5.3 The Global Grid Forum

The Global Grid Forum (GGF) was formed to bring Grid development together as 'an open process for development of agreements and specifications' and to provide 'a Forum for information exchange and Collaboration' [118].

The GGF therefore provides a structure for the discussion of standards and all of the research going on in the Grid community. This helps to identify key successful results and to keep everyone within the forum up to date with the developments.

The GGF also provides a steering group which is responsible for managing the GGF as well as handling a series of review documents following the progress of the development.

Within the GGF the research is divided into the following 7 areas, each of which is overseen by an Area Director within the GGF: Application and Programming Environments, Architecture, Data, Information Systems and Performance, Peer-to-Peer: Desktop Grids, Scheduling and Resource Management and Security.

At the beginning of 2003 there where 42 research groups involved with the GGF and this number is rising steadily. The GGF also hold regular meetings, which include tutorials, Workshops and Work sessions [118].

3.5.4 Middleware

The software to implement Grids is known as a Middleware because it is a middle layer which sits between the user's application and the remote computing resource.

It is generally accepted that the following key issues and services must be addressed within the Grid middleware, in order to be successful:

Networking Quality of Service (QoS), Resource co-scheduling, Load Balancing, Message Passing, File Transfer Mechanisms, Data Security, integrity and coherence and Authentication. [119] [120] [121] [122] [123] [124]

3.5.5 Existing Middleware implementations

There are several middleware implementations which exist although they all provide similar platforms [114].

Globus

Globus has become the standard for Grid middleware by providing all of the services and capabilities to construct a computational grid.

At the central core of Globus is the Globus Metacomputing Toolkit, which contains all of the tools needed:

Resource Management (GRAM) Resource allocation and process management.

Communication (Nexus) Unicast and multicast communication services.

Security (GSI) Authentication and related security services.

Information (MDS) Distributes access to structure and state information.

Health and Status (HBM) Monitoring of health and status of system components.

Remote data access (GASS) Remote access to data via sequential and parallel interfaces.

Executable management (GEM) Construction, caching and location of executables.

The reason for providing this selection of tools, is that computational grids must be able to support a vast range of different applications, and the services can be incorporated into the applications using a mix-and-match approach.

An important aspect of Globus is that it separates local and global services. Local services are kept simple to allow deployment and global services are built on top of local services.

The Metacomputing Directory Service is provided as part of the toolkit, to discover available resources and services. This allows resources to be added and removed dynamically and also allows the Grid to recover if a failure was to occur. By discovering the characteristics of the execution environment, it is able to automatically make the best choices for finding the most efficient settings [114].

UNICORE

The UNiform Interface to COmputing REsources (UNICORE) project, was initiated by the German Ministry for Education and Research (BMBF), with the aim of developing a middleware structure for ‘seamless access to supercomputer resources’ [125].

The main focus of their work is to provide a framework which will allow user’s, wishing to process large batch jobs, to seamlessly prepare, submit and monitor their jobs through an intuitive web based interface. One of the original goals was to ensure that UNICORE would be usable by scientists without having to know the anything specific about the middleware.

The emerging X.509 standard has been used as a method for authenticating user’s by the use of certificates. This also supports the encryption of data if it is to be transmitted over the internet.

Like most other middleware implementations there are a number of central components. These include: Job Creation, Job Management, Data Management, Application Support, Flow Control, Metacomputing, Interactive support, Single sign-on, Support for legacy jobs and Resource management [126].

The client software has been developed in Java, in an effort to try to make the software as platform independent as possible [125].

ICENI

The Imperial College e-Science Networked Infrastructure (ICENI) is the Grid middleware which has been developed at the London e-Science centre. It has been based upon the languages Java and Jini. Jini provides a service discovery architecture similar to Globus.

XML is used to store the attributes for each service, which are dynamically updated by the resource manager. A domain manager is provided which publicizes the resource capabilities within the Grid community.

Access control is implemented through a policy manager which controls who and what are allowed to use which resources. When resources are requested an X.509 public key infrastructure is used by the identity manager to verify users.

Other tools such as the resource browser, application mapper and resource broker are provided to access the resource information.

Legion

Legion is another Grid middleware which has been designed to connect many hosts together. The hosts can range from normal Desktop workstations to massively parallel supercomputers.

It is an object orientated environment providing a sound infrastructure for implementing a computational Grid. However unlike Globus the software is provided as a uniform programming model which can limit the programmer.

In order to be successful, Legion aims to satisfy the following objectives: Site autonomy, Support for heterogeneity, Extensibility, Ease of use, Parallel processing to achieve performance, Fault tolerance, Scalability, Security, Multi language Support and Global Naming [127].

Condor and Codine

There are other Grid middleware structures such as Condor and Codine which both provide similar features.

3.5.6 Existing Grid Projects

There are a number of existing Grid structures that have been implemented and are already being used for research.

UK e-Science Grid

The UK e-Science Grid, is the supercomputing backbone to the UK. It brings together all of the e-science supercomputing resources such as HPCx and HECToR. The HPCx Consortium [128] is run by the University of Edinburgh, providing further HPC resources with funding from the EPSRC. The *High End Computing Terascale Resource* (HECToR) [129] has recently been funded by the EPSRC, to provide a Cray XT4 supercomputer based at the University of Edinburgh's Advanced Computing Facility. Continuous funding has been allocated to guarantee updates in 2009 and 2011 ensuring that is will maintain its position as one of the most powerful research machines in Europe.

NASA's Information Power Grid (IPG)

NASA's Information Power Grid (IPG) is one of the largest grids in use today. It was set up to provide Supercomputing resources to provide NASA's research groups with the power they need when they need it. It is also the largest mass of freely available resources that any institution has made available to its researchers [130].

EuroGrid

The EuroGrid project was developed using funding granted by the Eurpoean Commision between November 2000 and January 2004. Its main objective was to establish and maintain a European GRID network consisting of HPC resources from centres in different European countries as shown in figure 3.15. Based on the UNICORE middleware, EuroGrid concentrated on several core applications, including bio-molecular modeling and weather forecasting as a basis to evaluate the usefulness of Grid resources. Although the project has now finished the resulting software is freely available for use in the research community [131].

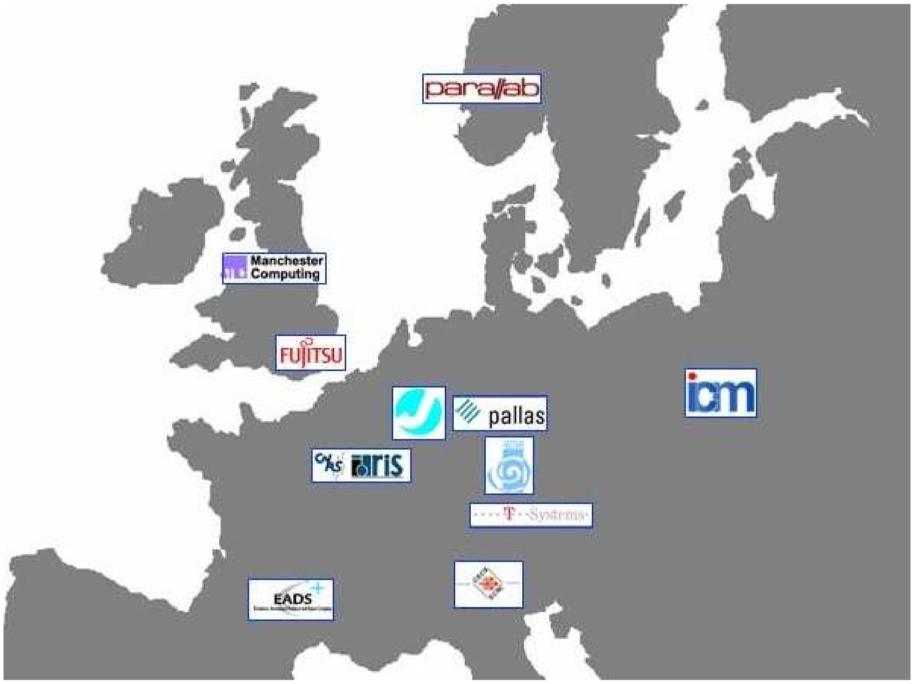


Figure 3.15: The EuroGrid Project established a grid of resources from centres in different European countries.

Particle Physics Data Grid

Particle Physics Data Grid Collaboratory Pilot (PPDG) developed and deployed a production Grid system with specific goals for modelling high-energy and nuclear physics experiments. By utilizing Grid resources significant increases were found in performance. One example of this comes from the Jefferson Lab for Nuclear Physics, who were able to reduce the time taken to simulate 30M events from over 3 months to less than a week using the PPDG [132].

TeraGrid

The TeraGrid is coordinated through the University of Chicago, and using high performance network connections, provides more than 750 taraflops of computing capability from across the US [133].

HealthGrid

The HealthGrid has been developed in order to meet the demands of modernizing health care systems. Grid computing provides an infrastructure allowing connectivity between health organizations and facilitates the secure sharing of patient records, research and information about new care methods. In particular the areas of where Grid technology has been most beneficial include:

- Medical imaging and image processing.
- Modelling the human body for therapy planning.
- Pharmaceutical Research and Development.
- Epidemiological studies.
- Genomic research and treatment development.

For all of the above areas, evidence has been produced that Grid technology can significantly reduce the cost and time taken to produce results [134].

3.5.7 AccessGrid

The AccessGrid is a very different use of grid resources. Instead of using the Grid infrastructure for computation, it uses audio and video streams to allow groups of user's to interact through a vast array of resources, including extensive video displays, shared presentation and interactive software environments. The main purpose of the AccessGrid is to allow 'large-scale distributed meetings, collaborative work sessions, seminars, lectures, tutorials and training' [135].

The main difference between the AccessGrid and existing desktop to desktop communication is that the AccessGrid focuses on groups interacting. For example by utilizing high end video and audio reproduction, the access grid allows groups of people to have the feeling of joining together in one large group meeting and thus increase productivity by removing time and travelling constraints.

However it is also possible to use the AccessGrid as a stand-alone desktop application. This makes it possible for a single user to join a larger group meeting, or even to have a one-to-one meeting. The real advantage of installing a personal AccessGrid node is that it can be installed on a standard computer system using very low cost cameras and microphones which are easily available. The personal node can then be used to sit in on meetings and distributed lectures for very low cost.

The AccessGrid software was originally developed and continues to be updated at the 'Futures Laboratory at Argonne National Laboratory (ANL) and is deployed by the NCSA PACI Alliance' [136].

The AccessGrid software is also highly configurable, allowing many different combinations of hardware to achieve the best and most appropriate solution for each venue. It is common for the software to be distributed across several machines locally to enable as many video and audio resources as required. For example, the software could be running on three computer systems each running a section of an extended video display. Then it would be logical for one of these systems to handle the video broadcast from your venue, another to handle the audio broadcast and the final computer could handle any shared applications.

Several projects have tried to integrate AR within the AccesGrid. VPC [137] is a replacement VideoProducer service for the AccessGrid which uses the ARToolkit to add virtual artifacts to the video stream of the presenter. Although the result is only presented in a 2D window, it allows the presenter to talk about various aspects of a virtual object by being able to rotate and move the object around by moving the ARToolkit marker. A similar application *Access Grid Augmented Virtual Environment* (AGAVE) [138] uses the AccessGrid to allow the presenter to share 3D content by capturing the scene with two cameras. The 3D scene is then presented to the audience who can either view it on any auto-stereoscopic style display. Generally when dealing with a large audience this would be done with passive 3D glasses.

3.5.8 Grid based HPV

It is possible to classify users of the Grid for HPV into four categories, as shown in Figure 3.16. Grid and Visualization specialists are the users who already have a working knowledge of the Grid, and Visualization and are generally the users who will continue to develop new Grid enabled Visualization tools. e-Scientists are users who have some familiarity with the Grid, but not visualization. Visualisation Specialists have knowledge of Visualization techniques, but not the Grid. Application Scientists have neither Grid or Visualization experience, however they form a large single group of potential visualization users. In the following list we explore some of the most common HPV systems and we also evaluate which user types they are most aimed towards:

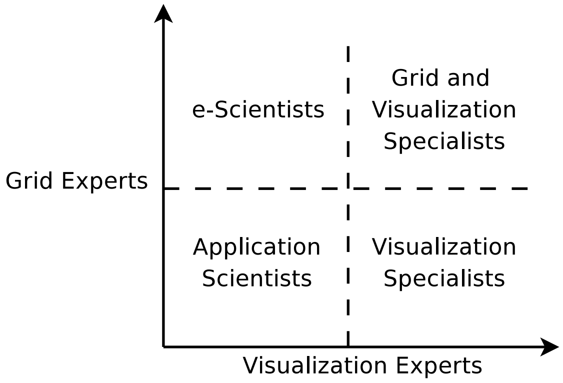


Figure 3.16: The user's of Grid based Visualization can be broken down into these four groups.

GVK

The Grid Visualization Kernel (GVK) [139] is designed to integrate the scientific visualization pipeline with Grid services. GVK monitors network conditions to dynamically modify the visualization pipeline, ensuring real-time performance to the user. GVK provides a modular approach to simulation and visualization severs, allowing for easy configuration for Visualization specialists who know how to con-Figure the visualization pipeline. Although initially designed to extend the OpenDX modular visualization system, interfaces are now provided for many other visualization environments, allowing the framework to be used arbitrarily with many software and hardware configurations.

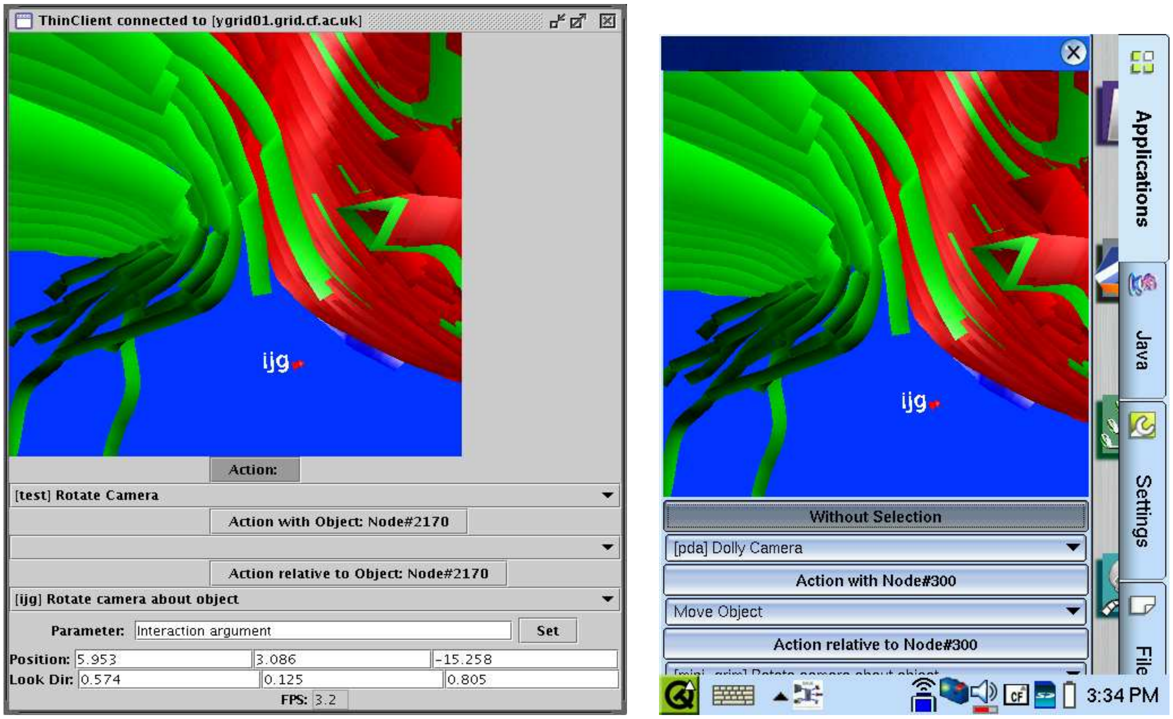
RAVE

The *Resource Aware Visualization Environment* (RAVE), [140] is a collaborative visualization environment which has been designed for deployment across a range of heterogeneous resources. RAVE allows multiple clients to connect and interact with a single visualization, regardless of the type of client that they are using. It is a unique framework, in that it supports a wide range of clients from large scale displays to PDA's and mobile phones which have very limited graphical capabilities for local rendering.

RAVE is seen as *resource aware* because the format of the graphics sent to each client is determined by the capabilities of the client and supporting networks. A remote render service is provided for any client which is unable to locally render the scene, and will sent the graphics to the client at a compressed image stream. If the client has limited graphical capabilities then the dataset is likely to be decimated to a size that can be rendered locally. Figure 3.17(a) shows an active client which is running on a Linux PC and connected to a simulation. The Linux PC has some basic graphics capabilities and so the rendering is being done locally. Figure 3.17(b) shows a thin client running on a PDA. It is connected to the same simulation, except in this case it has no graphical capabilities and so a remote render service is being used to generate the graphics, which are being streamed to the PDA as compressed images.

Chromium

Chromium [141] works differently to most visualization platforms. It works by capturing the OpenGL stream generated from the mapping stage of the visualization pipeline and distributes it over a series of processing nodes. Each node renders a


ThinClient connected to [ygrid01.grid.cf.ac.uk]
ijg
Action:
Action with Object: Node#2170
Action relative to Object: Node#2170
Set
Without Selection
Action with Node#300
Action relative to Node#300
Applications
Java
Settings
File
3:34 PM

[test] Rotate Camera

[ijg] Rotate camera about object

Parameter: Interaction argument

Position: 5.953 3.086 -15.258

Look Dir: 0.574 0.125 0.805

FPS: 3.2

[pda] Dolly Camera

Move Object

(a)

(b)

Figure 3.17: An example of the RAVE client running on (a) A Linux PC (b) A Windows CE based PDA, where both clients are connected to the same visualization. Images courtesy of Ian Grimstead, Cardiff University.

section of the final image and these are all composited to produce the final image. By its nature it is very easy to integrate Chromium with any application which generates an OpenGL stream making it compatible with a large number of visualization applications. Previous work at Bangor University [142] has extended Chromium to integrate with the Globus Toolkit, is shown in Figure 3.18.

SGI Vizserver

SGI developed Vizserver [143] to enable their range of Onyx Servers to be used as multi-user visualization servers. Vizserver provides a thin client which decompresses

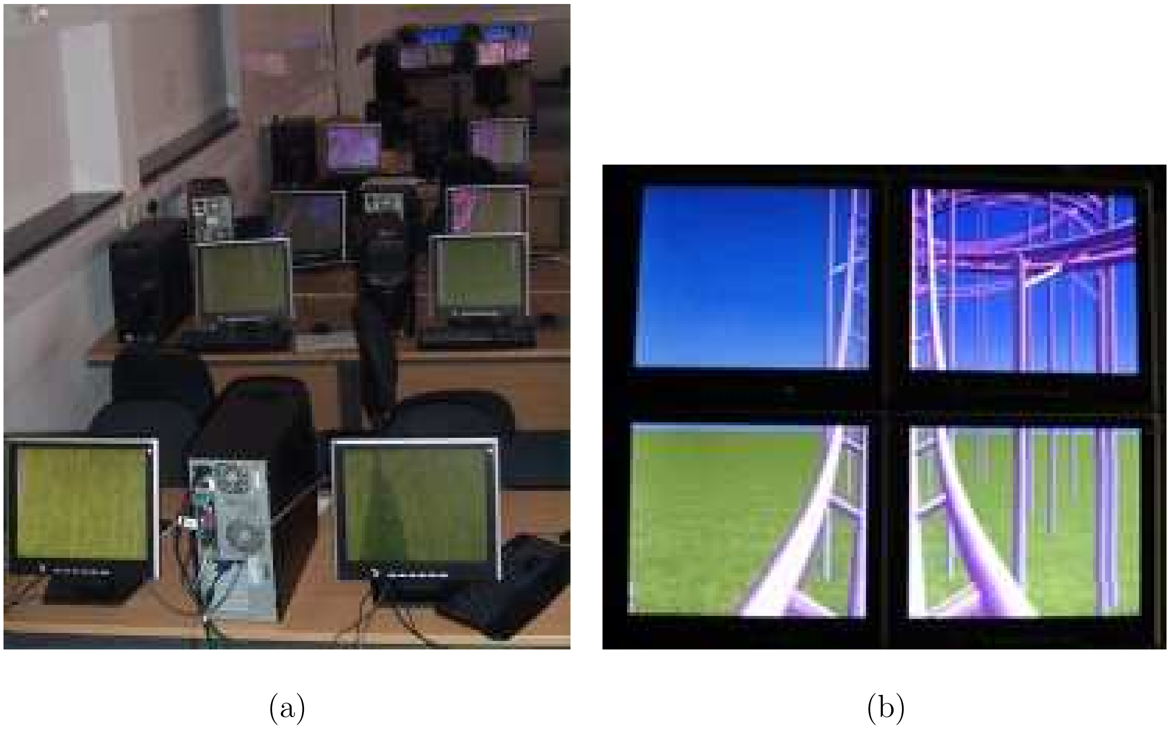

(a) (b)

Figure 3.18: An OpenGL demonstration is distributed with Chromium (a) across 16 desktop computers and (b) tiled across a 2x2 display. Images courtesy of Ade Fewings.

and displays a stream of images that have been rendered remotely, bringing the power of the Visualization server directly to the user's desktop. Vizserver is currently only compatible with SGI's rage of servers and does not support connections through Grid middleware.

The Op3D system [144] was designed to utilize the visualization capabilities of a *Shared Memory Parallel* (SMP) machine using integrated graphics hardware and was specifically implemented using a high end SGI Onyx machine. It is primarily designed to volume render data and return a stream of frames to the client machine which provides a simple viewer interface, allowing surgeons, who are Application Scientists, to view and interact with large 3D datasets of medical scans whilst working in an operating theatre. Op3D uses OpenGl Volumizer to provide high quality rendered images, and SGI Vizserver to compress and transfer the images to the operator's laptop.

Visapult project

Visapult [145], is also used for remote rendering of large datasets. It provides scalability by distributing the volume dataset over several disks, and therefore transferred to the visualization servers in parallel. Visapult has been specifically designed for handling time-stepped volume data and therefore is unsuitable for general use by application scientists. Due to the parallel design of the data handling however, it does become very efficient for volume rendering extremely large datasets, and has been demonstrated with datasets over 10GBs.

gViz

The gViz project [146] provides a collaborative Grid visualization platform based around the modular system IRIS Explorer. Modules can be configured to form pipelines, using the supplied dataflow network editor, allowing the construction of almost any type of visualization. As knowledge of visualization techniques is still needed to construct the pipelines, the software is aimed primarily at Visualization specialists.

During development of the gViz project a reference model was suggested for describing visualization tasks, as shown in Figure 3.19. At the conceptual layer the visualization task is described independently of the hardware and software required to support it. The logical layer, adds details of the software used for the visualization and the physical layer describes the underlying hardware [146].

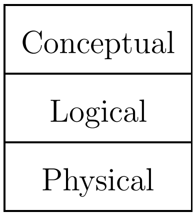


Figure 3.19: The gViz reference model for describing visualization tasks.

e-Viz

The e-Viz system [147] was developed as a visualization tool for application scientists, following the Autonomic Visualization model in order to attempt to make choices on behalf of the user. It is based around a framework of pre-defined visualization services, each of which can be implemented using different configurations of visualization system, such as VTK.

e-Viz follows the principles of the gViz reference model where the user is provided with a number of suitable options for rendering their visualization task, specified at the conceptual level. The process of specifying the task at the logical and physical levels is left to the e-Viz system. This means that the user only has to choose the desired type of visualization and not have to understand how or where it is implemented.

The e-Viz remote rendering library enables multiple visualization servers to return frames to a single client, using several codecs for compressing the data. By interchanging the compression method with different codecs, e-Viz is able to balance network bandwidth and image quality to ensure real-time performance. The client is also able to change between each of the visualization server streams during run-time and servers can be added and removed at any point during a visualization session, allowing seamless migration between servers. This allows for e-Viz to give the user a low quality visualization whilst waiting for more powerful machines to become available. It also allows for mission critical applications, where two streams can be synchronized and the client can dynamically switch should one of the servers fail.

The list of possible rendering options is identified by a decision making module that uses a knowledge-base to see what configurations have been used successfully for similar tasks previously. A simulation framework, SimuVis [148] is used to simulate the performance of different visualization pipeline configurations allowing e-Viz to dynamically optimize the pipeline during the visualization run-time.

e-Viz provides both an API for developers to write their own software, as well as a wizard style launcher application. The wizard identifies the user's data set, and having generated a list of possible configurations, presents the user with a thumbnail to represent how each configuration may look, to assist them with starting their visualization session.

The development of the e-Viz framework was fundamental to the research described in the thesis. This was a challenging three year project using resource and expertise from four universities. Bangor university contributed significantly to the implementation of the e-Viz broker and database components, as well as the overall deign and testing of the framework [27] [147]. This thesis contributes to the exemplar applications of the framework and the development of the framework.

3.5.9 Web Services and the Grid Summary

The Grid provides a middleware infrastructure for allowing resources to be used which are not necessarily local to the user in geographical terms. It allows each user to effectively have HPC resources at their desktop, by making the resources available transparently. Several Grid implementations have been developed following guidelines set by the GGF.

Several toolkits have been developed that use the Grid as a mechanism for bringing HPV resources to the user's desktop. In the context of visualization, this chapter showed that there are four categories of Grid Based Visualization user: e-Scientists, Application Scientists, Grid and Visualization Specialists and Visualization Specialists. Each of the toolkits has been developed and targeted at either e-Scientists or Visualization Specialists, who have some specialist knowledge. e-Viz, a project which was developed in parallel with this research, was developed to target Application Scientists, using an autonomous approach to helping make decisions for inexperienced user's.

3.6 Grid enabled AR?

This review identifies several approaches to AR, although the most successful of these rely upon markers to align the virtual artifacts with the real world. More sophisticated approaches are possible but they are not widely used due to the inherent computational overhead typically involved. There has also been very little research done into using HPV resources to render the virtual objects within the AR scene and to use the pose estimation to steer the virtual camera in the remote simulation. Chapter 5 presents our unique approach to AR, by extracting feature points from the user's view to allow us to estimate the movement of any arbitrary object in the real world. Chapter 6, shows how the e-Viz framework can be used to remotely render large data-sets which form the virtual objects in our AR scene. To ensure real-time performance this thesis also shows how the pose estimation component of our AR system can be distributed to a remote computational resource.

Chapter 4

Optical Tracking for AR

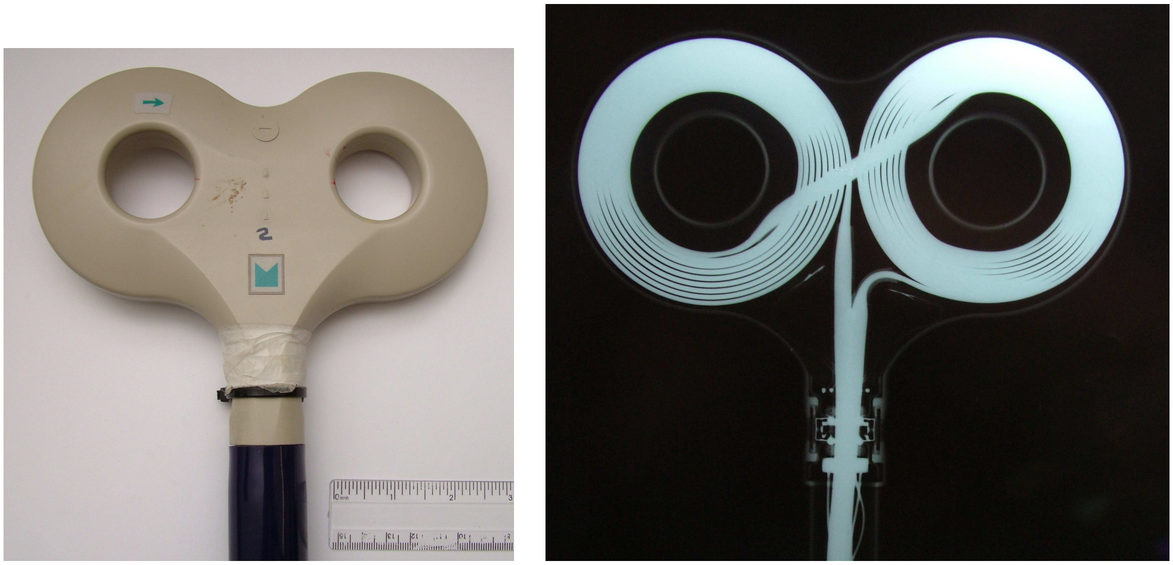
4.1 Introduction

In order to investigate the research hypothesis a challenging AR application was selected. Transcranial Magnetic Stimulation (TMS), is a procedure in which electrical activity in the brain is influenced by a pulsed magnetic field [149]. TMS is extremely important for researchers as it allows them to accurately stimulate different regions of the brain's cortex and by recording the subject's response it is possible to validate the function of different areas of the brain. TMS has also been found to be useful in therapy and has had positive results when attempting to treat severe depression, auditory hallucinations and tinnitus as well as other drug resistant mental illnesses such as epilepsy.

Common practice during the TMS procedure is to place an electromagnetic coil on the subject's head so that it is aligned with a region of interest on the cortex of their brain. The coil and subject's head can be tracked using optical sensors, and targeting information is calculated and displayed on a local workstation. This procedure allows analysis of the visually induced perceptions related to the cortical site stimulated.

Typical TMS installations utilize a Polaris optical tracking system, as developed and manufactured by Northern Digital Incorporated (NDI), to track the patients head and to help align the electromagnetic coil with the regions of interest on the surface of the patient's cranium.

This chapter describes the software prototype developed for interfacing with the Polaris optical tracking system that allows users to find the quaternion and translation position of each of the tools. A custom OpenGL application is then used to allow the operator to identify and specify regions of interest in the cranium. The software also guides the operator to accurately align the coil with the specific regions and then uses VTK to render the patient's brain according to the operator's viewpoint, allowing for both an offset calibration for the tools and real-time movement of the patient and operator.



(a) (b)

Figure 4.1: A typical electromagnetic coil (a) and the magnetic field generated (b) as used for TMS.

TMS has an important role in neuroscience as it allows researchers to understand which regions of the brain are actually used to perform different tasks. Previously it has been possible to use noninvasive techniques such as fMRI to allow researchers to identify which regions of the brain are active whilst the subject is asked to perform specific tasks. However TMS has now been proved to show that although an area of the brain is active with a task it does not necessarily show that it controls the task.

4.1.1 Clinical uses for TMS

TMS has been shown to have successful results when used for both diagnostic and therapeutic uses:

Diagnostic Uses

TMS allows clinicians to meanure the activity and function of specific parts of the human brain. To date the most widely accepted use is for measuring the strength of a connection between the primary motor cortex and different muscles within the body. This makes TMS a useful diagnostic and assessment tool for cases involving strokes, spinal cord injury, multiple sclerosis and motor neuron disease [150].

Therapeutic Uses

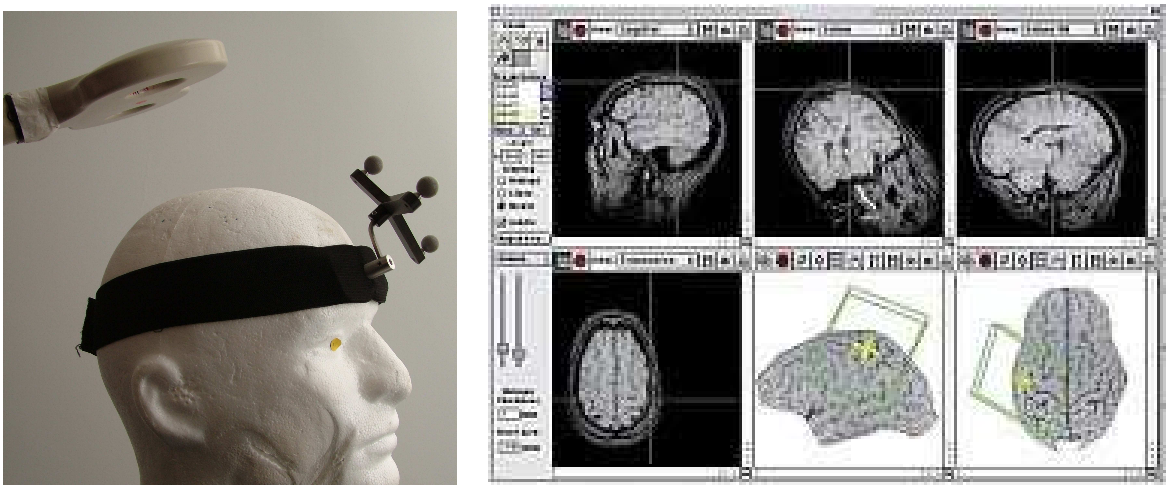
Herrmann and Ebmeier used TMS to show that by exciting specific neurons in the cortex of their trial subjects, they were able to treat them for depression [151]. Based upon this previous work the American Psychiatric Association launched the NeuroStar TMS TherapyTM system in 2006, a clinical trial designed to evaluate TMS as a real world treatment for several depressive disorders.

Further research has shown TMS to be an effective method for treating migraines as well as for controlling sleeping patterns for subject's who are being treated whilst asleep [152]. TMS has therefore an established treatment for patients suffering from the above problems. Any improvements that can be made to the treatment environment can only be beneficial.

Other conditions which have been reported to respond to TMS include: Tinnitus, Parkinson's Disease, Dystonia, Epilepsy, Migraine, Dysphasia and Hemispatial neglect [153].

4.2 Brainsight Software

The Brainsight, frame-less image-guided, TMS system (Magstim, UK) used during this project, incorporates a Polaris optical tracking system for tracking the electromagnetic coil and its relative position to the subject's head.

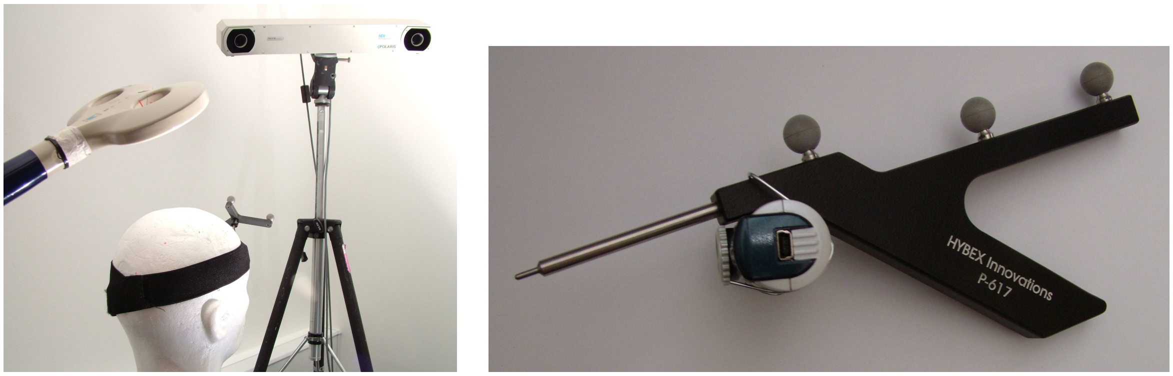


(a) (b)

Figure 4.2: (a)The tracker used to identify the position of the subjects head, and (b) The Brainsight software displaying targetting information on a computer display.

4.3 Optical Tracking

In order to accurately render the patients brain from the view point of the operator, the system needs to know the orientation of both the subject's head and the view point of the operator. The Polaris Optical tracking system (described in Chapter 3) is supplied with several tools, which include a head position tracker and a digitizing probe that is used for the accurate pinpointing of positions in the trackers view. The head position marker is used to represent the position of the subject's head and the probe to identify the position of the operator's viewpoint - see Figure 4.3.



(a) (b)

Figure 4.3: The tools being tracked: (a) The subjects head, The electromagnetic coil and (b) the viewpoint of the operator captured using a webcam aligned with the digitizing probe

4.3.1 Polaris Interface

Communication between the operator's laptop and the Polaris System is achieved by sending serial ASCII text messages using the RS232 standard as shown in Figure 4.4. All communications are initiated by the laptop, which sends a message and will in response receive a reply either containing the requested information or a flag

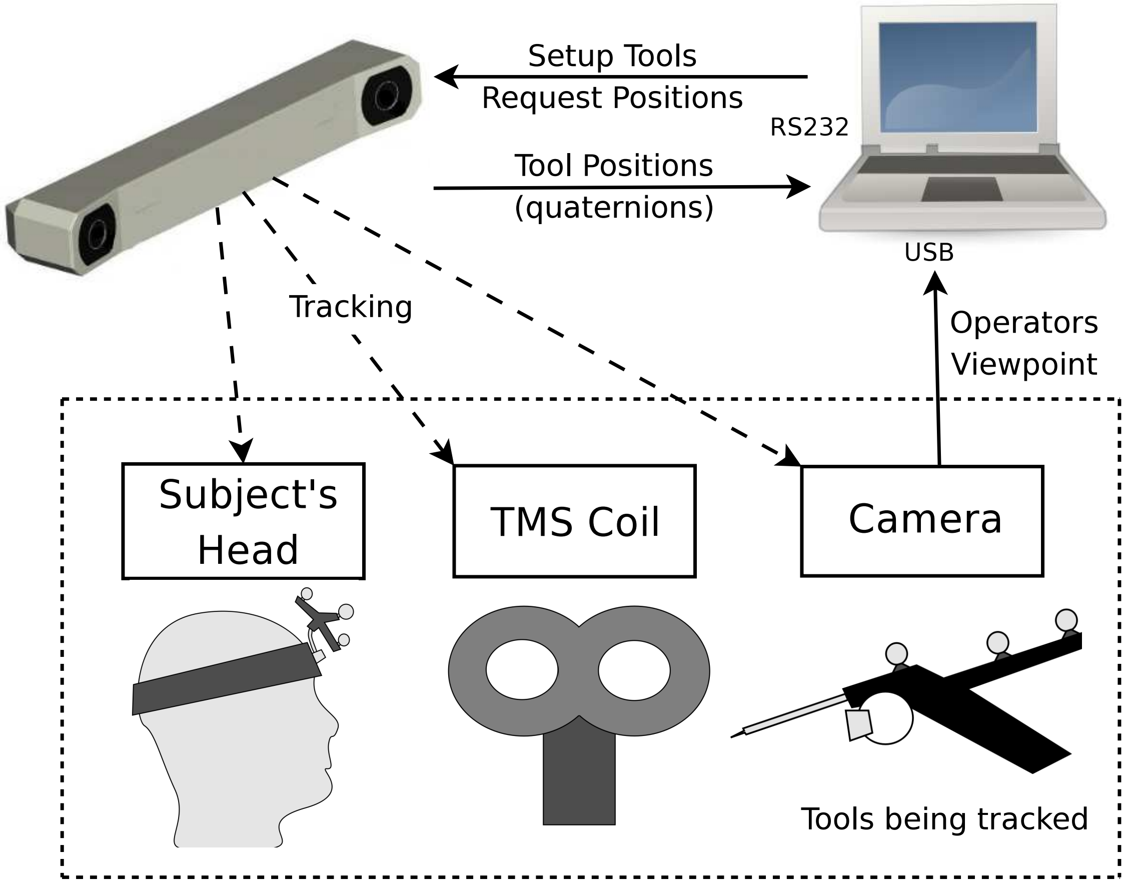


Figure 4.4: System Diagram showing the flow of data and the tools that are being tracked.

indicating whether or not the last command was successful. Messages sent to the Polaris system always take one of the following two formats:

```
Command | <SPACE> | Parameter  1 | Parameter  2 ... | Parameter  n | <CR>
Command | <:> | Parameter  1 | Parameter  2 ... | Parameter  n | CRC16 | <CR>
```

The Polaris system always responds with the following format:

```
Reply | CRC16 | <CR>
```

The description of the marker configuration for each tool is specified in a ROM file, which is provided by the manufacturer, and each tool is *plugged* into the tracker by passing the contents of these ROM files in messages. It is then possible to simply query the tracker as to the position of each tool, which it returns as a quaternion and translation pair.

4.3.2 Error Checking

A 16-bit Cyclic Redundancy Check (CRC) based upon the IBM standard is used to validate the integrity of each command sent to the Polaris system and each reply received from the Polaris system, using the polynomial:

$$x^{16} + x^{15} + x^2 + 1$$

Therefore should an error occur our software can detect it and either retry the previous command, or return an error to the operator saying that an error has occurred. The Polaris system also provides the ability to search the surrounding area for environmental IR sources which can cause interference to the marker reflected IR and therefore cause inaccurate results. The operator can then either identify and remove these IR sources or the software can compensate for the possible loss of precision [154].

4.4 Video Capture

The software that has been developed is named *Bangor AR for TMS* (BART). In BART the operator's viewpoint is captured by a USB web-cam, using *video4linux (V4L)* which provides the video capture/overlay framework API for the linux kernel.

The V4L API provides a generic interface to USB and Firewire based imaging devices [155]. BART allows the user the flexibility to use any general purpose webcam and allows our software to capture an image from the camera at each time-step of our applications execution. V4L also follows the same conventions of the *video4windows (V4W)* API allowing that the software could be ported to the Microsoft Windows platform with ease.

4.5 Coordinate system

BART uses a 3D Cartesian coordinate system which follows the right hand rule for the viewing transformations, specifying the directions of positive and negative as shown in Figure 4.5(a). This is the same convention as used by both OpenGL and VTK, allowing our transformation information to be moved easily between the graphics API's being used.

To allow us to realistically match the virtual world to the real world a Perspective Projection is used. This allows the objects in our virtual environment to be *projected* onto a viewing plane. This plane can then be matched to the view captured by the camera view point. Although sometimes in computer graphics, it would not be suitable to use Orthographic Projection as an alternative. Orthographic Projection is used mainly for design applications where the z-axis coordinate is ignored. This would mean that each virtual object would be the same size regardless of their distance from the camera [156].

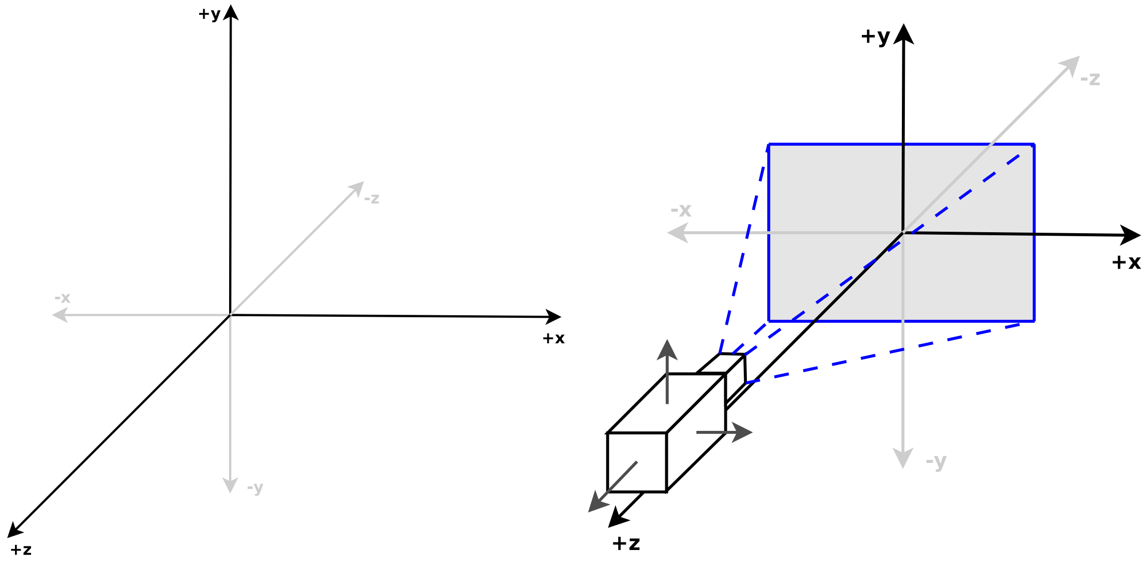


Figure 4.5: The virtual Camera, which is aligned with the real camera from the real world, showing the viewing plane in Cartesian space.

4.6 Standardizing the Coordinate System

In order to correctly align the subject's brain the position of each tool is requested from the tracker and returned in quaternion and transformation form.

Q_0, Q_x, Q_y, Q_z represent the quaternion rotation of the tool and T_x, T_y, T_z represent the translational components of the transformation. To enable the rotation to be used in our OpenGL application, the quaternion representation is converted into a rotation matrix [157]:

$$M = \begin{pmatrix} M_a & M_b & M_c \\ M_d & M_e & M_f \\ M_g & M_h & M_i \end{pmatrix}$$

$M_a = (Q_0Q_0) + (Q_xQ_x)(Q_yQ_y)(Q_zQ_z)$

$M_b = 2((Q_xQ_y)(Q_0Q_z))$

$M_c = 2((Q_x Q_z) + (Q_0 Q_y))$

$M_d = 2((Q_x Q_y) + (Q_0 Q_z))$

$M_e = (Q_0 Q_0)(Q_x Q_x) + (Q_y Q_y)(Q_z Q_z)$

$M_f = 2((Q_y Q_z)(Q_0 Q_x))$

$M_g = 2((Q_x Q_z)(Q_0 Q_y))$

$M_h = 2((Q_y Q_z) + (Q_0 Q_x))$

$M_i = (Q_0 Q_0)(Q_x Q_x)(Q_y Q_y) + (Q_z Q_z)$

The Transformation Matrix can then be derived by combining the translation information with the rotation matrix M:

$$M_1 = \begin{pmatrix} M_a & M_b & M_c & T_x \\ M_e & M_f & M_g & T_y \\ M_i & M_j & M_k & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

We can simply transform to the position of the subject's head and draw the brain, then transform to the operator's viewpoint using an offset established during calibration to produce the final viewpoint. This view is then composited onto the video stream from the camera and presented back to the operator via the HMD.

4.7 Calibration

In order to correctly align the computer graphics with the real world, it is essential to calibrate the BART software to understand the offset between the tracked position of the physical markers and their relation to the real world position of the objects.

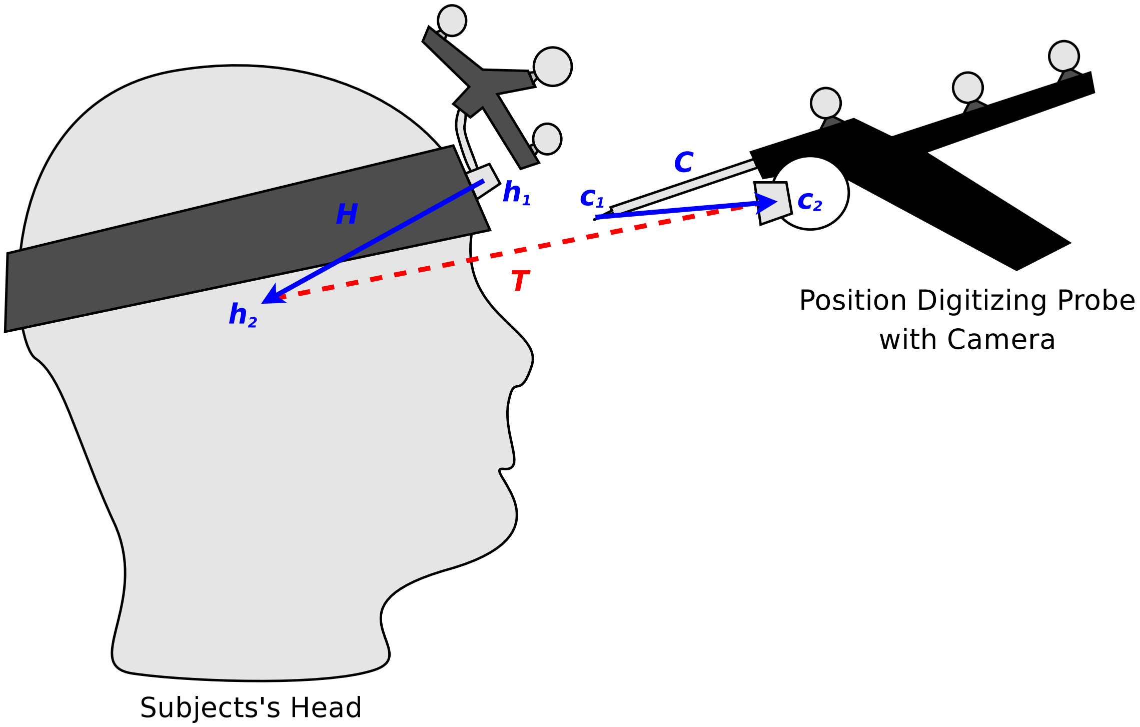


Figure 4.6: The Calibration offset for both the Camera C and the subject's head H need to be calculated.

Figure 4.6, illustrates the offset transformation which is needed by the markers on both the camera and the subject's head. The Digitizing probe used to specify the position of the camera is designed to track the position at c_1, however we need to calculate the relationship between this point and the lens of the camera at c_2. Likewise the tracker used to track the subject's head only gives the software the position h_1 and therefore we will need to calculate the difference between h_1 and h_2.

4.7.1 Camera Position Calibration

The calibration of the camera's position is calculated using a *static point*, such as a pencil taped to a desk as a reference. As shown in Figure 4.7, the Digitizing probe is firstly aligned with the static reference and its position stored as transformation matrix C_1. The Digitizing probe is then moved forward to align the camera lens with the reference point and the second position is then stored as C_2. In order to correctly align the virtual camera with the real world camera the transformation matrix could be evaluated using the calibration transformation matrix C, in the following way:

$$C_1 \times C \times C_2$$

Therefore the transformation matrix C representing the relationship between the probe and the camera can then be calculated as:

$$C = C_2^{-1} \times C_1$$

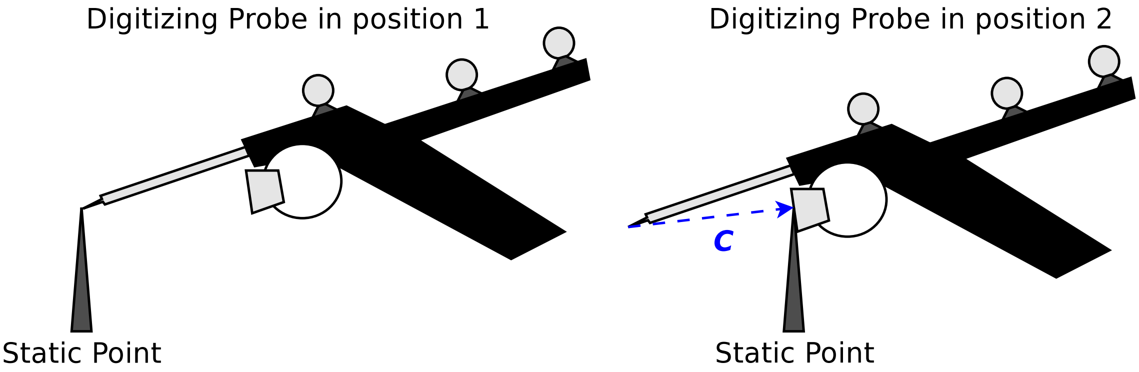

Digitizing Probe in position 1
Digitizing Probe in position 2
C
Static Point
Static Point

Figure 4.7: The relationship C between the Digitizing Probe and the camera lens is calculated using a static point as a reference.

4.7.2 Head Position Calibration

The Head Position Calibration, as shown in Figure 4.6 as H, is specified manually by the operator. Whilst the application is running, the operator has the option to manually change the alignment of the virtual cranium as an overlay of the actual subjects head. This allows for movement of the brain to be taken into account and for the operator to correct any alignment problems whilst using the software.

4.8 Targeting

The BART AR Interface has been implemented in order to make alignment of the TMS coil with the specific regions of the subjects cranium. To do this the operator must first specify the regions on the surface of the cranium using our software to identify the areas that they wish to target. The targeting tool then guides the operator to accurately align the TMS coil with the specified area.

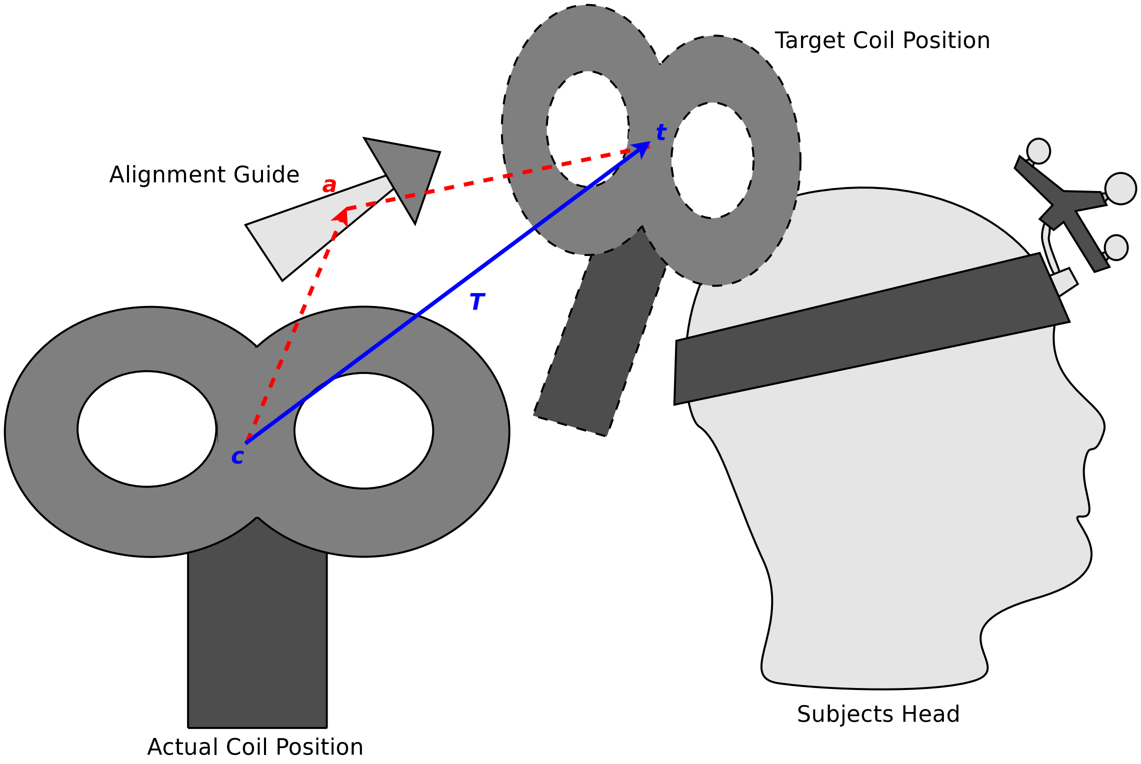


Figure 4.8: The transformation is calculated between the actual position of the coil and the target position of the tool.

The operator is guided by a 3D arrow which is rendered from their viewpoint, as shown in Figure 4.8. The transformation of the arrow is calculated by firstly calculating the transformation between the actual position of the coil and the target position T, where the translation from the origin to the actual position is T_1 and the translation from the origin to the target position is T_2:

$$T = T_2^{-1} \times T_1$$

The translation from T_1 to the guide arrow A is then calculated in the following way:

$$A = T - T_1$$

A second OpenGL window is used to display only the guide arrow as shown in Figure 4.9. The arrow is rendered from the viewpoint of the operator as before except the translation information is disregarded, allowing the operator to see the guide arrow full screen. This is calculated by taking only the 3x3 Rotation component from the transformation matrix:

$$A = \begin{pmatrix} A_a & A_b & A_c & A_d \\ A_e & A_f & A_g & A_h \\ A_i & A_j & A_k & A_l \\ A_m & A_n & A_o & A_p \end{pmatrix} A_1 = \begin{pmatrix} A_a & A_b & A_c & 0 \\ A_e & A_f & A_g & 0 \\ A_i & A_j & A_k & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4.9 Rendering the Graphics

The cranium is previously segmented from the MRI data captured for the specific patient. The BART software uses VTK to render the cranium using the standard

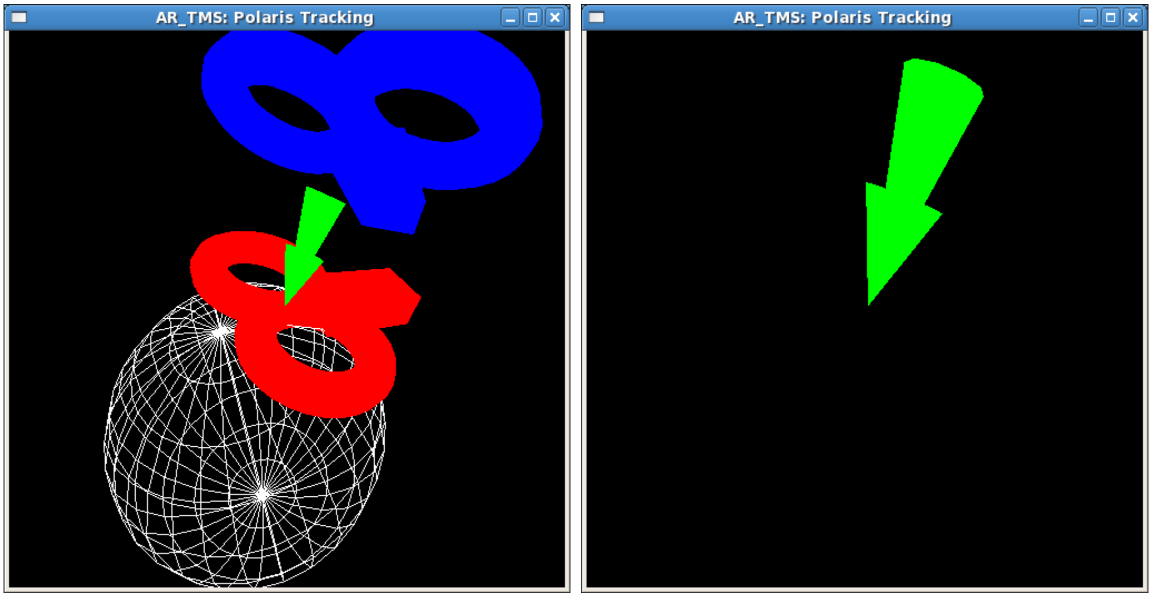


Figure 4.9: The OpenGL Targeting interface guides the operator to align the magnetic coil with the desired region of interest.

visualization pipeline as shown in Figure 4.10, using a single transformation matrix as shown in Figure 4.6. Whilst the rendered display is updated in real time by the position of the subject's head, and position of the camera the operator also has the opportunity to fine tune that calibration by manually updating the offset matrix for the tracker on the subject's head.

4.10 Results

As shown in Figure 4.11 the alignment is visually acceptable, and the AR interface for positioning the coil is more natural than having to look between the subject and the results displayed on the workstation monitor (even if the workstation display is also projected onto the wall).

However there are some limitations as to how well the software can perform.

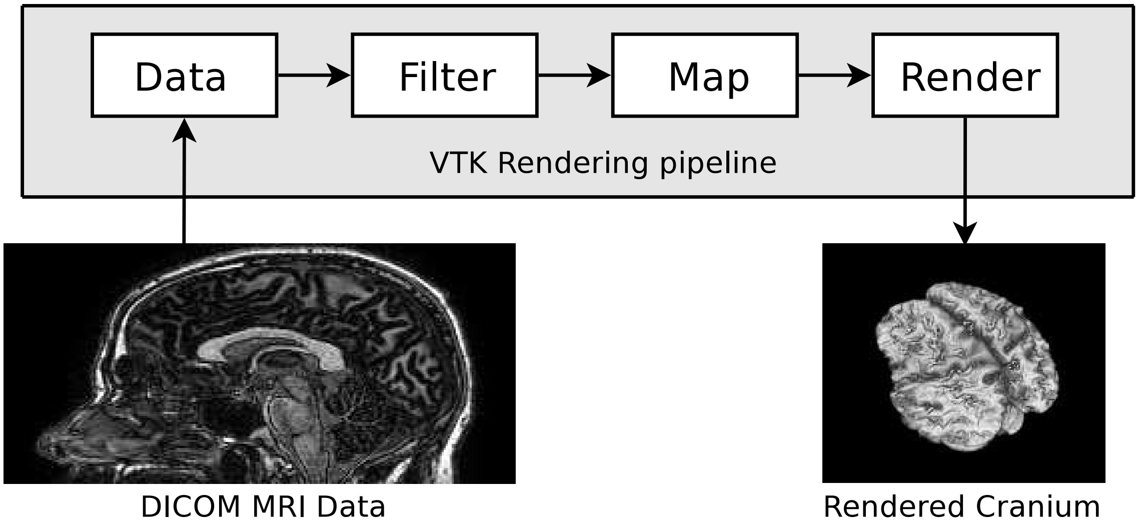


Figure 4.10: The VTK rendering pipeline.

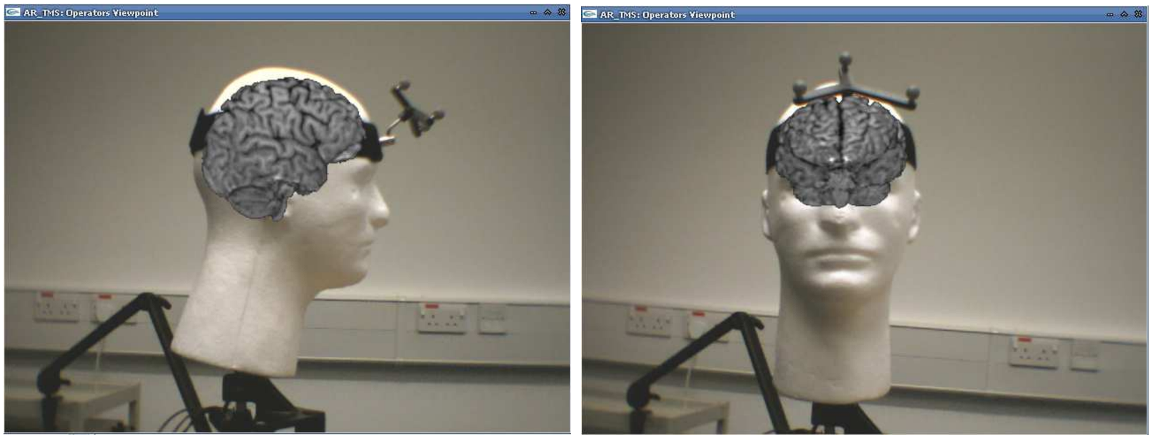


Figure 4.11: The composited operator's view showing a visually acceptable alignment of the real world and the computer graphics

The laptop computer only has basic graphics capabilities and although is capable of rendering the subjects brain at an average 15FPS, it can only render the graphics at a low screen resolution, whilst maintaining this real time performance. It is also not yet possible to composite the rendered cranium into the user's viewpoint without vastly reducing the FPS and losing the realism of the AR environment. However these issues are addressed and solutions are found in subsequent chapters.

4.11 Conclusions

Although many AR applications currently rely upon the use of bold markers that can be easily identified by a computer, it is becoming increasingly desirable to be able to use AR in environments where there are no clear markers. Markerless tracking is not only more computationally intensive however, but also requires more information about the environment and the structure of any planes or real objects that are to be tracked.

Due to the obvious dangers involved with modifying the natural activity inside a patients's brain, there is a clear need for validation to ensure that alignment is correct between the patient and the rendered cranium. This will ensure that the clinician is actually targeting the intentional region of the brain within an acceptable tolerance. It is very difficult to measure the accuracy of the alignment, particularly due to the both the manual calibration which requires the cranium to be aligned visually by the operator and the natural movement of the cranium within the skull. The alignment is always going to be limited by the ability and experience of the operator and it can be expected that there will be a significant difference between the calibration performed by both an expert and a novice. Any validation studies would need to ensure that the calibration stage was included and would also need to take into account a range of different calibrations performed by operators of different experience.

There are several approaches that could be taken to validating the accuracy of the alignment. It would be possible to run both BART and the existing Brainsight Software, which has been generally accepted to be accurate, and any discrepancies between them should be observed. Another approach would be to create a phantom head, which after calibration could be disassembled and internal components could probed, to confirm their actual position. Fischer et al, used similar validation study to validate MRI image overlay for needle puncture alignment [158].

In order for BART to be a useful tool several metrics must be defined. These must include quantifiable allowances such as a tolerance for how accurately the coil must be aligned. Although a full validation study is outside the scope of this project, favourable comments about the ease of use of the AR interface have been obtained from the TMS operator's in the School of Psychology at Bangor University.

The next chapter presents a generic solution that does not rely on the use of markers, but rather feature points that are intelligently extracted from the user's view.

Chapter 5

Using Feature Point Extraction for Pose Estimation

5.1 Introduction

In order to align the virtual object with the real world, a developer to define the object in the user's view. During a calibration stage the user is given the opportunity to specify where the object exists within the viewpoint. This work uses a robust feature point detection algorithm to identify the points that can be repeatedly identified within the space occupied by the virtual object and use this information to estimate and track the object's pose (i.e. position and orientation) as it is moved from its initial position. A Cascading Haar-Classifier [159] is used to track the position of the object within the user's viewpoint. A POSIT (Pose from Orthography and Scaling with ITeraions) algorithm is then used to estimate the pose of the object [160] .

In order to use the feature point based tracking within BART v2, the tracking system is extended to use multiple calibrated views. A Haar-Classifier is then used to rapidly match the user's current view with each of the stored calibrated views.

5.2 Computer Vision

Computer vision techniques are used to enable the computer to actually ‘see’ what the user is looking at. There are many camera options available to capture an image data stream of the surrounding environment, but this work uses a cheap proprietary web-cam which is connected to the Linux workstation. Also the Open Source Computer Vision Library (OpenCV) from Intel is used to capture and manipulate the image data stream from the camera using the *Video 4 Linux* API, see Figure 5.1.

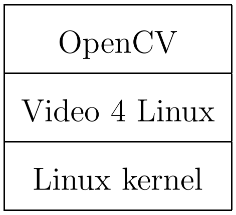


Figure 5.1: The software layers for Computer Vision

During this research the user’s viewpoint was captured using a Sony Eye-toy USB camera, capable of providing a stable 640x480 pixel resolution at 30 hertz, with a 56 degree field of view. The Sony Eye-toy uses the standard OV7640 compatible sensor, which can be decoded by *Video 4 Linux* using the open source OmniVision OV5xx drivers developed by Mark McClelland [161]. This camera was chosen because it offered a high specification in an inexpensive and easily available camera.

5.3 Calibration

In a similar approach to the software developed in chapter 3, a calibration stage is used to record the transformation relationship between the virtual objects and their position in the real world. BART v2 initializes by asking the operator to align a basic wire-frame model with the position that it relates to in the view captured by the camera representing the operator's viewpoint. Basic mouse interaction is used to align the virtual object with the real world, and the operator is given the opportunity to store multiple viewpoints with the correctly aligned virtual object. The software relies upon the ability of the computer vision system to identify the real world representation of the virtual object, so by capturing multiple views it is possible to store calibration information for different views of the object. Figure 5.2 shows the calibration tool being used to register a virtual object with a real world object.

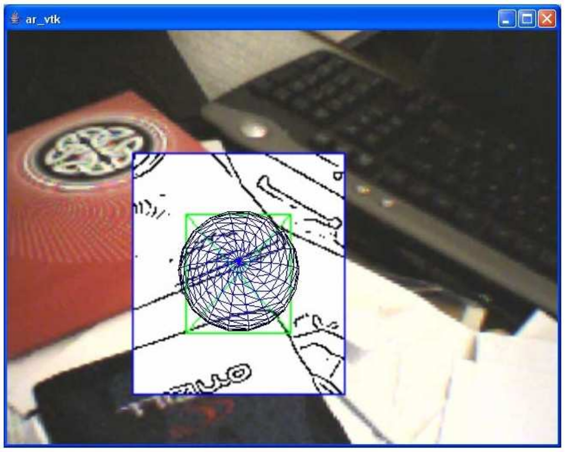


Figure 5.2: The Calibration tool allows the user to align a virtual object with an object in the real world.

The calibration information is stored as a combination of the transformation matrix representing the position that the operator has moved the virtual object to from the origin, and the subset of the image data which contains the object in the real world. Later the subset of the image data is converted into a set of feature points, which can be used to calculate the final pose of the object.

5.4 Object Tracking

BART v2 application uses the calibration that links our virtual object to a space in the operator's real-time view, to train a Haar-classifier [159]. The Haar-classifier is then used to estimate which sub image from our operator's view contains the real object, to which a virtual object is to be aligned. This work extends previous work with Haar-classifiers by using multiple calibration views allowing the detector to not only be more robust but also to continue to track different sides of an object.

5.4.1 Haar-classifier

Haar-classifiers use a set of simple geometric shapes, known as Haar-like features, which are used as building blocks for specifying more complex shapes. This gives many advantages over using raw pixel data as Haar-like features can be used to encode simple image data and provide a domain knowledge specification that can be used to specify whether two patterns match. Each Haar-like feature has a value, which is defined as the difference between the total number of black pixels and the total number of white pixels. This makes the classification tolerable to environmental changes, such as light because the value is calculated as a ratio between the black and white pixel areas.

Each Haar-like feature consists of several white and black regions. Figure 5.3 shows an example set of these features although it is much more common to use an extended set that was proposed by Lienhart and Maydt [162].

Learning

In order for the Haar-classifier to be able to identify accurately the where the object is likely to be within the operator's viewpoint, it is necessary to train the classifier

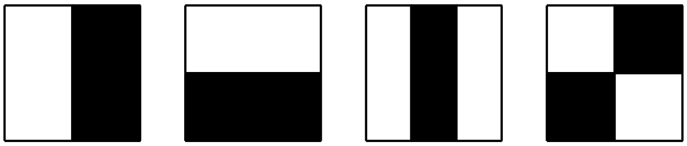


Figure 5.3: An example set of Haar-like features.

with source images. The OpenCV toolkit is used to generate a series of training images by scaling, rotating and adding noise to the calibration images. A second set of random noise images are also generated as negative training examples.

The AdaBoost algorithm [163] is used to train the classifiers. Initially each training sample is allocated an initial weight. The first of our classifiers is set to use the single Haar-like feature that achieves the most reliable recognition of each of the training images. This produces a *weak* classifier as it will still produce a high rate of inaccurate results. Each of the training samples that produced an incorrect recognition is then given a higher weighting, a further classifier is then set to use a more detailed Haar-like feature having tested it against each of the higher weighted images. This process is then repeated to produce a linear set of *weak* classifiers which are designed to work together to produce the accuracy of a *strong* classifier.

Classification

The operator's viewpoint is split into a series of subsections, each of which is applied to the cascaded Haar-classifier. If the subsection is within a threshold of the value of the first classifier, then it will propagate through each classifier until either it fails to match a classifier or it passes all of the classifiers. When a subsection is deemed a failure it is instantly discarded and the next subsection is classified. If a subsection passes then it is classified as containing the object and it is stored with a likelihood weighting. Once each of the subsections has been tested the highest weighted subsection is declared the winner and will now be comparable to our classification image for pose estimation as shown in Figure 5.4.

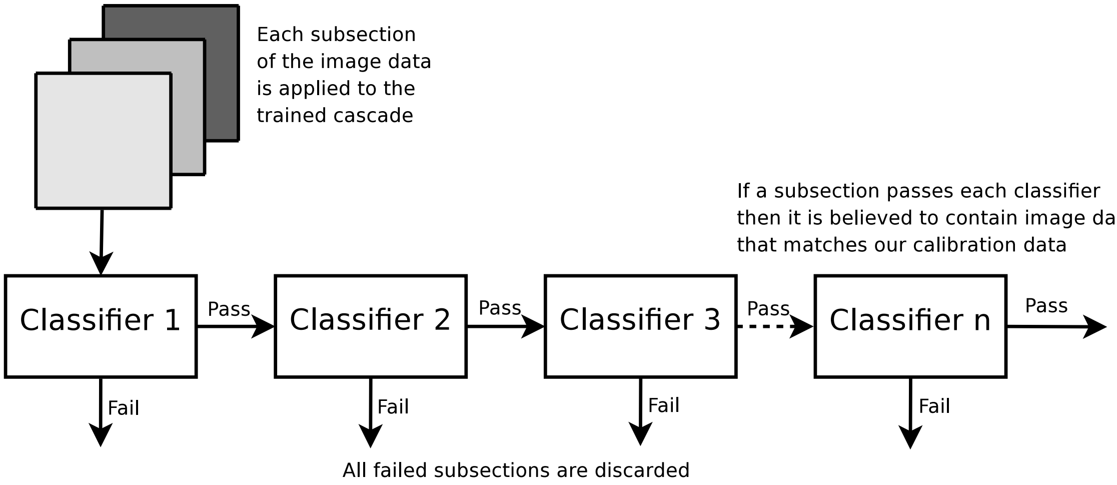

Each subsection
of the image data
is applied to the
trained cascade
If a subsection passes each classifier
then it is believed to contain image data
that matches our calibration data
Classifier 1
Pass
Classifier 2
Pass
Classifier 3
Pass
Classifier n
Pass
Fail
Fail
Fail
Fail
All failed subsections are discarded

Figure 5.4: The cascaded Haar-classifier is presented with subsections of the operator's viewpoint. The subsection propagates through each classifier until either it passes each stage or it fails a single classifier, at which point the subsection is immediately rejected.

5.4.2 Parallel Classification

To allow the software to use multiple calibration views multiple cascaded classifiers are used to classify each of the different calibrations. When each classification is complete the highest weighted subsection out of all of the classifiers is declared the winner and in this case both the subsection and the transformation matrix that relates to that view are passed to handle the pose estimation. The BART v2 software uses separate threads for each cascaded classifier to utilize multiple processors where available, as shown in Figure 5.5. Although this approach has increased the robustness of BART v2, the computational requirements of the software has also increased.

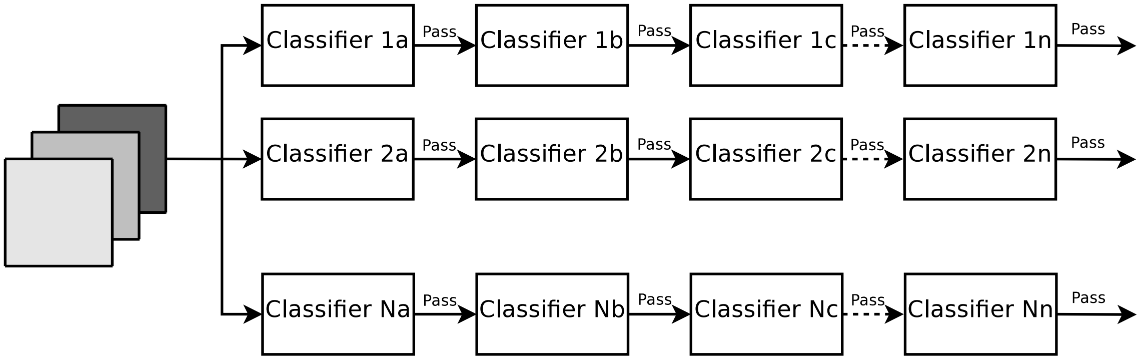


Figure 5.5: Multiple cascaded classifiers are used in parallel in order to identify the source of the subsection from each of the different calibrated views.

5.5 Pose Estimation

The pose of a 3D object can be described by using a combination of its transformation and rotation. DeMenthon and Davis developed a POS (Position from Orthography and Scaling) algorithm for estimating the perspective projection with a scaled orthographic projection and therefore calculates the transformation matrix for the specified object. Consequently the POS algorithm is used to compare the feature points extracted from the relevant subsection of calibration view to feature points extracted from the section of operator's current viewpoint that has been classified by the Haar-classifier as containing the actual object that the system is tracking, as shown in Figure 5.6 [160].

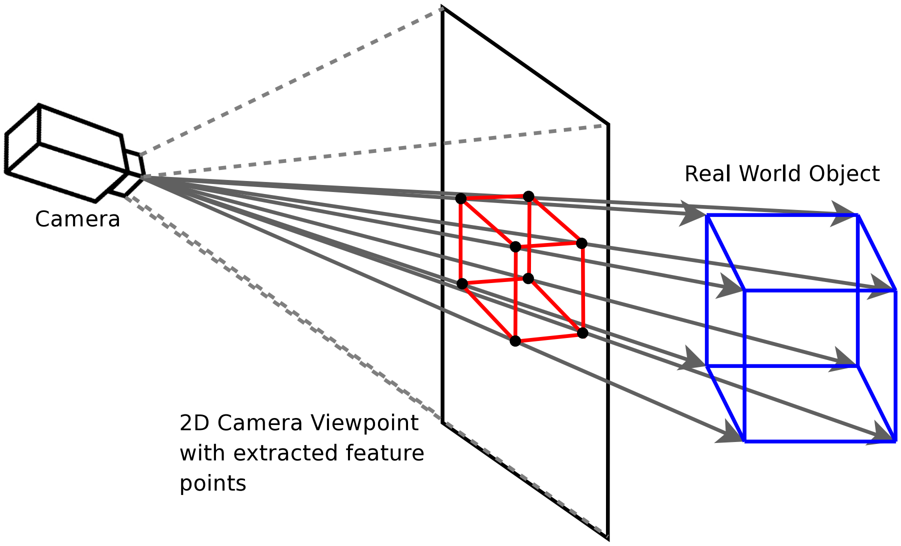


Figure 5.6: The POS algorithm maps the 2D feature points from the operator's viewpoint to the real world 3D object.

5.5.1 Implementation

OpenCV provides an implementation of the POSIT algorithm with the function *cvPOSIT()*. The function takes an array of two dimensional feature points from both the operator's viewpoint and the calibrated image and returns a 4x4 matrix representing the OpenGL transformation matrix. By simply applying the POSIT algorithm to a 2D representation of the calibration image we eliminate the need to store 3D geometric information about our virtual object and simply calculate the difference. As shown in Chapter 3, the final transformation is calculated by combining the POSIT transformation P with the calibration transformation C to calculate the total transformation T:

$$T = C^{-1} \times P$$

5.6 Results

Figure 5.7 shows the complete system diagram for the software that has been developed for the generic AR framework. In order to fully evaluate the framework an updated version of the TMS interface, which no longer requires specific markers has been used. A short video clip (100 frames) of the camera panning around the subjects head was recorded to evaluate our software. Four views of the subjects head where used to calibrate BART as shown in Figure 5.8. The test video was deliberately designed to progress around the subject's head clearly including three close matches to the calibrated views.

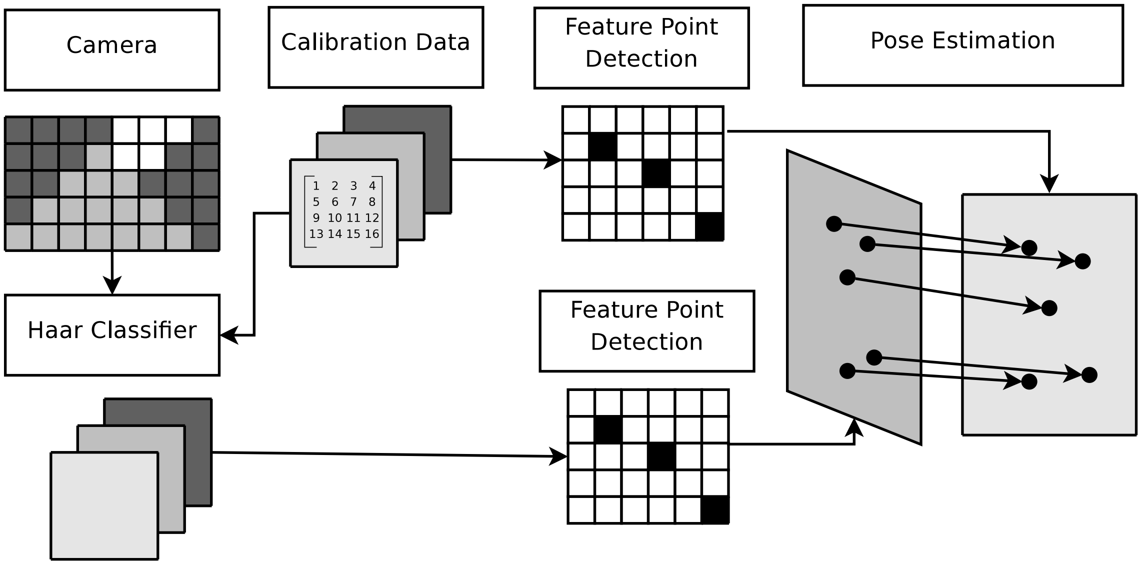


Figure 5.7: The complete system diagram for our software.

A desktop PC (Athlon 3200 64Bit, 1GB RAM) was used to run the BART v3 software. Figure 5.8 shows an example frame from the evaluation video. The red box signifies which of the four calibration images has been selected by the Haar-classifier, and the blue lines illustrate some of the feature points which have been matched and used to estimate the pose of the subjects head. Although not drawn whilst the software was running the subject's cranium has later been composited into the frame, using the transformation matrix that was calculated for the frame during run-time. The subject's cranium appears to line up correctly with the subject's head showing an visually acceptable alignment.

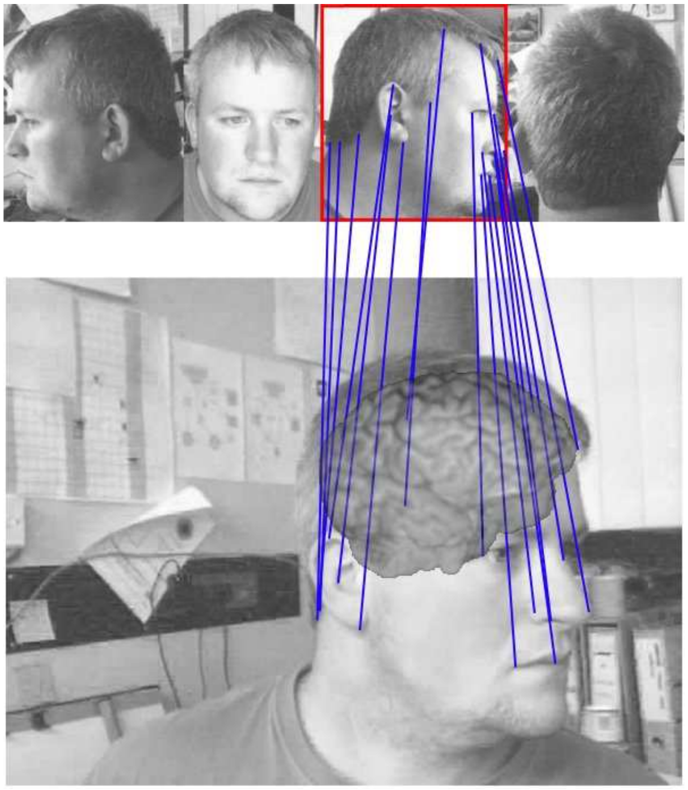


Figure 5.8: A frame from our evaluation video, showing which calibration image has been used and some of the feature points that have been matched to it, whilst calculating the pose.

Figures 5.9 - 5.12 show the output of the four classifiers that have been trained and applied to our test video. Each classifier generates a rating for each frame with a value between 0 (unlikely to contain the pattern) and 1 (most likely to contain the pattern). It is clear to see that around frames 25 and 65 there is the most confusion about which classifier has matched the pattern. This is because those frames represent the boundary between two calibrated images, therefore it is possible that both classifiers will have found a match. The fourth classifier at no point features in the test video and so has the lowest weighting for each frame. The application was found to be robust over an extended period of time, however only 100 frames are shown for illustrative purposes.

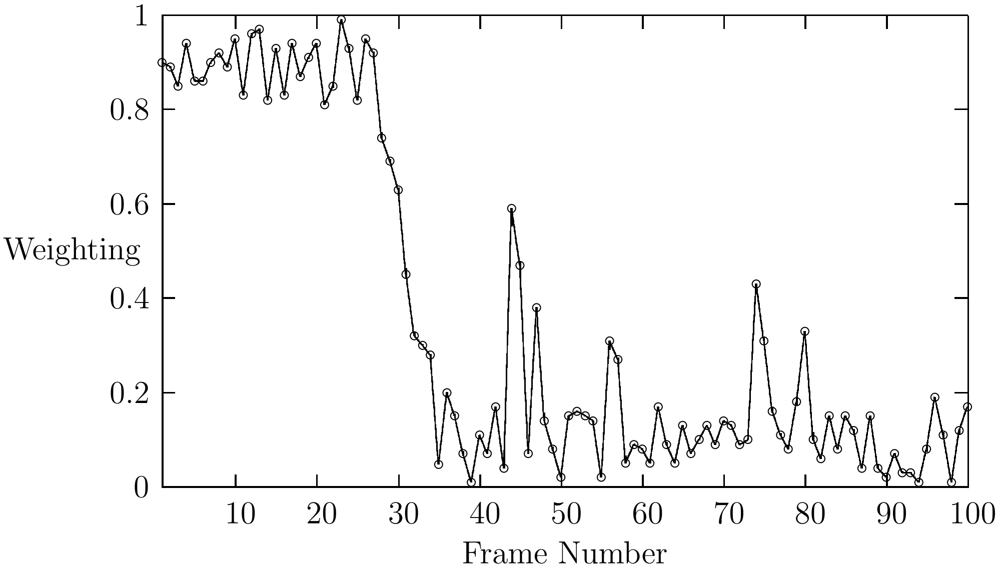


Figure 5.9: The weighting that classifier (1) allocated to each image of the test video.

Figure 5.13 shows the time taken to estimate the pose for each frame. We can see that although the alignment has been accurate it is currently far from real-time taking between 180ms and 500ms to calculate each. This gives us an average frame rate of 5.49FPS, which is a significant drop from the 15FPS that, as stated in Chapter 1, is required to maintain the illusion of AR in real-time. It is also possible to see a significant increase in the time take at point (a) and (b). This is most likely because as the camera approaches the border between to calibrated views it is becoming more difficult to classify the image.

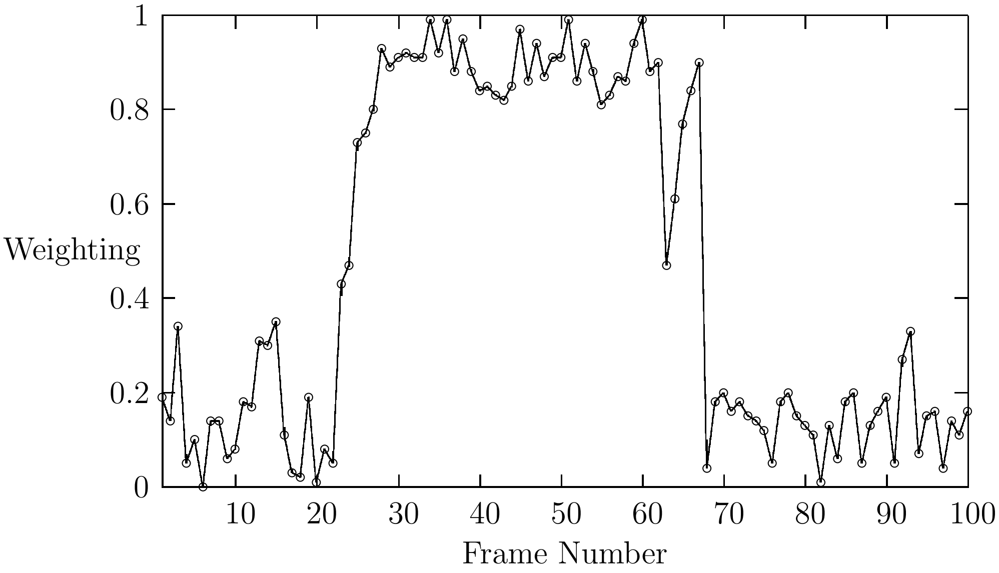


Figure 5.10: The weighting that classifier (2) allocated to each image of the test video.

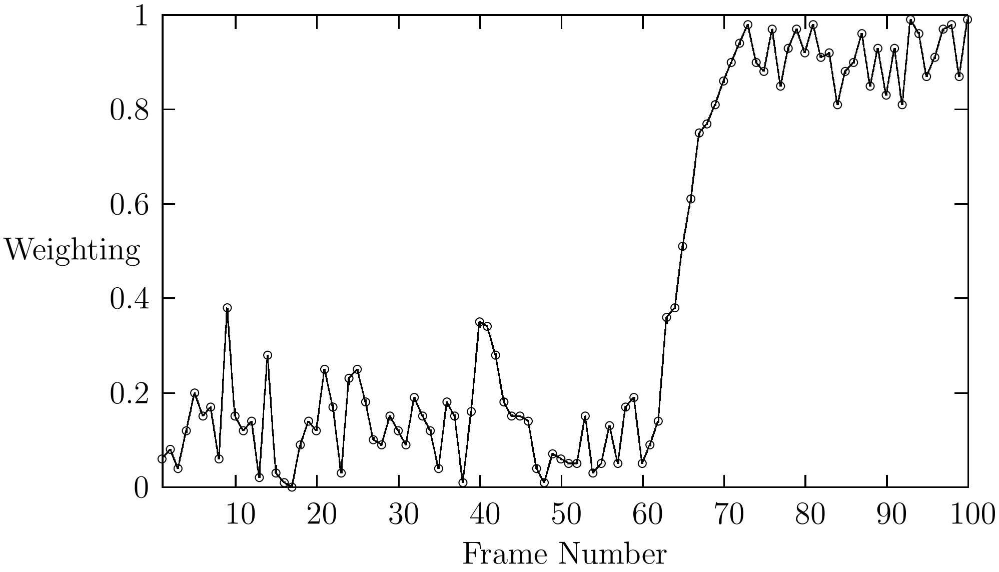


Figure 5.11: The weighting that classifier (3) allocated to each image of the test video.

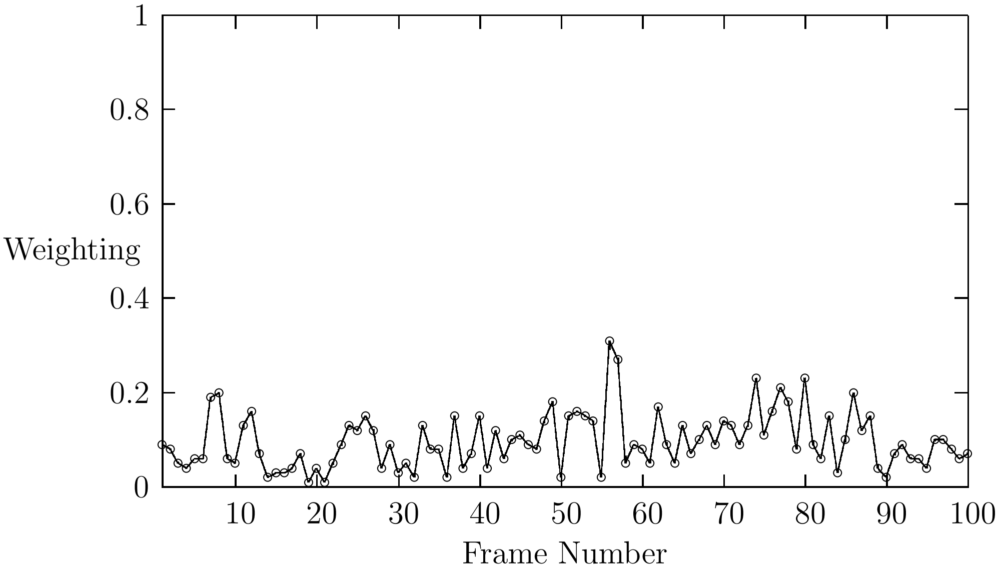


Figure 5.12: The weighting that classifier (4) allocated to each image of the test video.

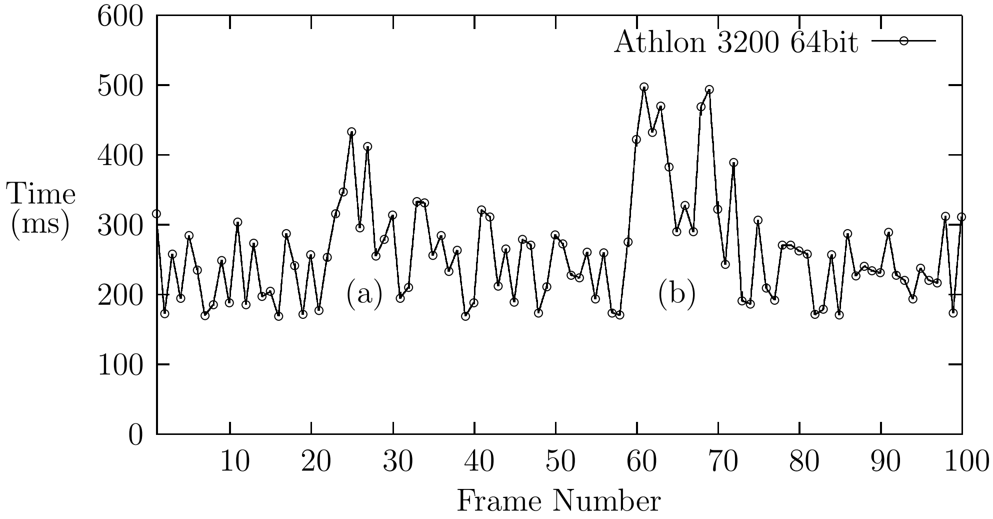


Figure 5.13: This graph shows the time take to process each frame of a pre-recorded video stream.

5.7 Conclusions

This work demonstrates that it is possible to provide a framework for AR without markers provided that environmental conditions, such as lighting, remain constant. In varying lighting conditions the software is unable to accurately estimate the pose due to a poor recognition of the repeatable feature points. This is not generally a problem within TMS laboratories, where the lighting remains constant however this would make our software unsuited to outdoor applications where light levels constantly change. Gausemeier et al, provide a solution to this by using low-pass filters to attempt to separate the image data from the environmental light [164]. This technique could be used to filter the camera image data and to remove the effects of the lighting. Another approach that has been found to be successful is to extract repeatable image data through a histogram analysis [165].

Problems also occur when trying to run the AR software on a local workstation. It is simply too computationally intensive and so can not keep up with the real time video. The next chapter explores how this problem can be solved by distributing this processing to a more powerful remote Grid resource.

Chapter 6

Grid enabled Augmented Reality

6.1 Introduction

The developed BART v2 framework for AR, which although has been shown to be accurate in alignment, has so far been unusable due the computational requirements hindering the real-time performance. In this chapter we explore the how real-time performance can be achieved by exploiting remote Grid enabled computing resources, using the e-Viz framework. Firstly demonstrating how the e-Viz framework can be used to increase the quality of the rendered objects by distributing the rendering process to a remote visualization server. Second demonstrating how the AR application, BART v3, can be used successfully used as a real-time interface to TMS by distributing the pose tracking and estimation to a remote resource as part of the e-Viz visualization pipeline.

The e-Viz system [147] was developed with funding from the EPSRC and was a collaborative project involving computer scientists at Bangor, Swansea, Leeds, and Manchester universities. The project developed a framework that aims to address some key issues with visualization on the Grid. Although as seen in Chapter 2, projects such as GVK and gViz already offer an architecture for visualization on the Grid, e-Viz attempts to abstract away from the underlying implementation and provide a ‘black box’ where the user has no need to consider the underlying implementation details unless they already have specialist knowledge in this area.

Extending Haber and McNabb’s classical visualization pipeline ideas [32] on to the computational Grid allows many of the features of the Grid to be inherited by the visualization system. The use of distributed computation and parallel remote rendering allow visualization of data sets of increasing size, which can be rendered at high resolutions whilst still maintaining interactive frame rates through the use of parallelism at each stage of the visualization pipeline. The Grid framework can also provide many other features such as remote data acquisition and security based upon the Grid Security Infrastructure (GSI).

In order for a Grid based visualization system to be truly generic it must be able to support any visualization technique that the user requires. e-Viz is able to dynamically select the most appropriate hardware and software for each visualization task as well as providing system transparency to the user. The Grid can then be seen as a ‘black box’ inside which the computation is done. The user should not be concerned about what happens inside the ‘black box’ with only the result that it produces.

6.2 Adaptive Visualization

The e-Viz framework aims to follow IBM’s *Autonomic Computing* approach [65], as described in Chapter 2, in order to intelligently choose the most appropriate software and hardware for each visualization task.

6.2.1 Towards Autonomic Computing

Figure 6.1 presents a functional description of the e-Viz framework, relating each of the coloured components to the adaptive layer of the deployment model. For each *User Visualization problem* the framework provides the user with a series of options that it believes are most suited for running the visualisation task. A simulation of each of the available hardware and software combinations is performed in the *System and Task Simulation* service layer, using SimuVis [148] to find an optimized visualization pipeline for each visualization task. The Knowledge Server stores information about how different pipelines can be configured and this information is used to generate a *Formal Pipeline Description*. The User interface is created based upon the *Formal Pipeline Description* allowing the user to adjust each component of the visualization pipeline. During run-time the visualization pipeline and hardware and software choices can be adapted to better meet the user's requirements.

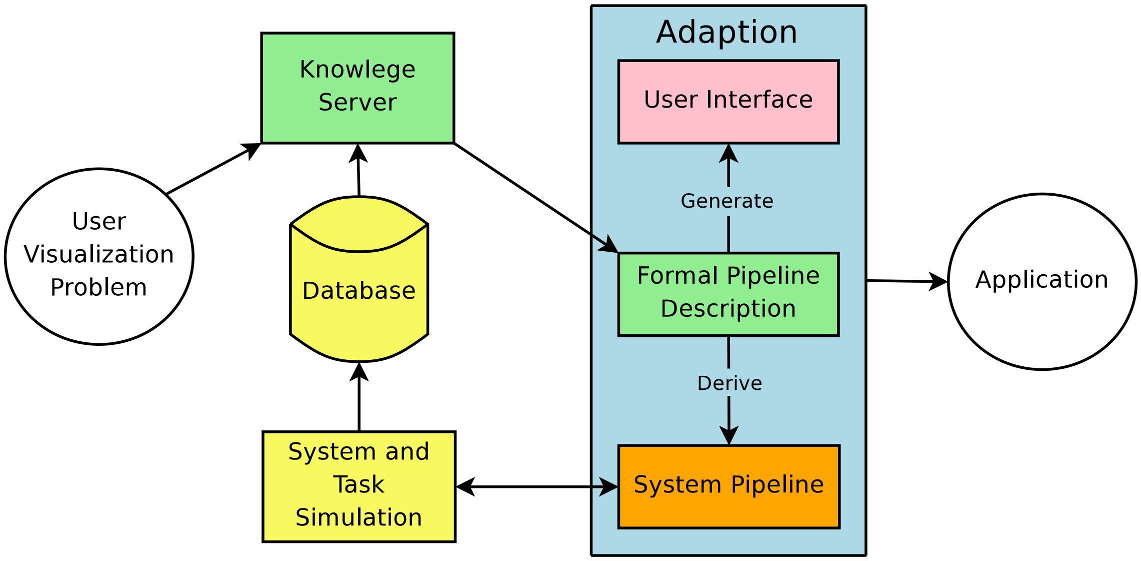


Figure 6.1: The functional description of the e-Viz framework. Each of the coloured components relates to its appropriate level of the same colour in Figure 2.9.

6.2.2 SimuVis

SimuVis was developed as part of the e-Viz system to allow the simulation of autonomic visualization systems. SimuVis was based on SimEAC [148] and provided a design tool allowing for the rapid prototyping and evaluation of the components of the e-Viz framework. It also forms a large component of the *Decision Making Module*, which is able to evaluate different hardware, software and visualization pipeline configurations to find the one most suited to the visualization task. Chisnall gives an extensive review of the functionality of the *Decision Making Module* [148].

SimuVis allows the underlying infrastructure of a visualization system to be specified as a collection of hardware and software attributes which as within in an XML document. The simulation then evaluates the XML description and provides statistics for the performance of the system. By evaluating the system 'offline' users are able to save the cost of expensive CPU-time as well as avoid configurations which could result in a failure of a shared service system. The *Decision Making Module* continues to evaluate different configurations during a running visualization. It is therefore able to dynamically switch between resources, in order to self-optimize the visualization task.

6.3 Grid Visualization with e-Viz

The e-Viz framework consists of three main components as shown in Figure 6.2. These are described below:

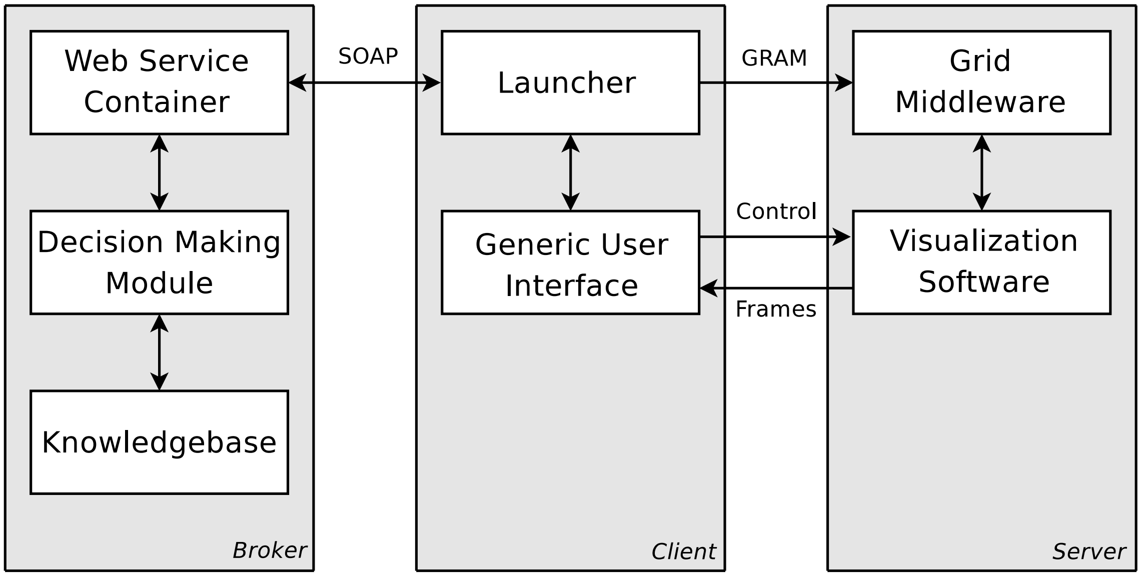
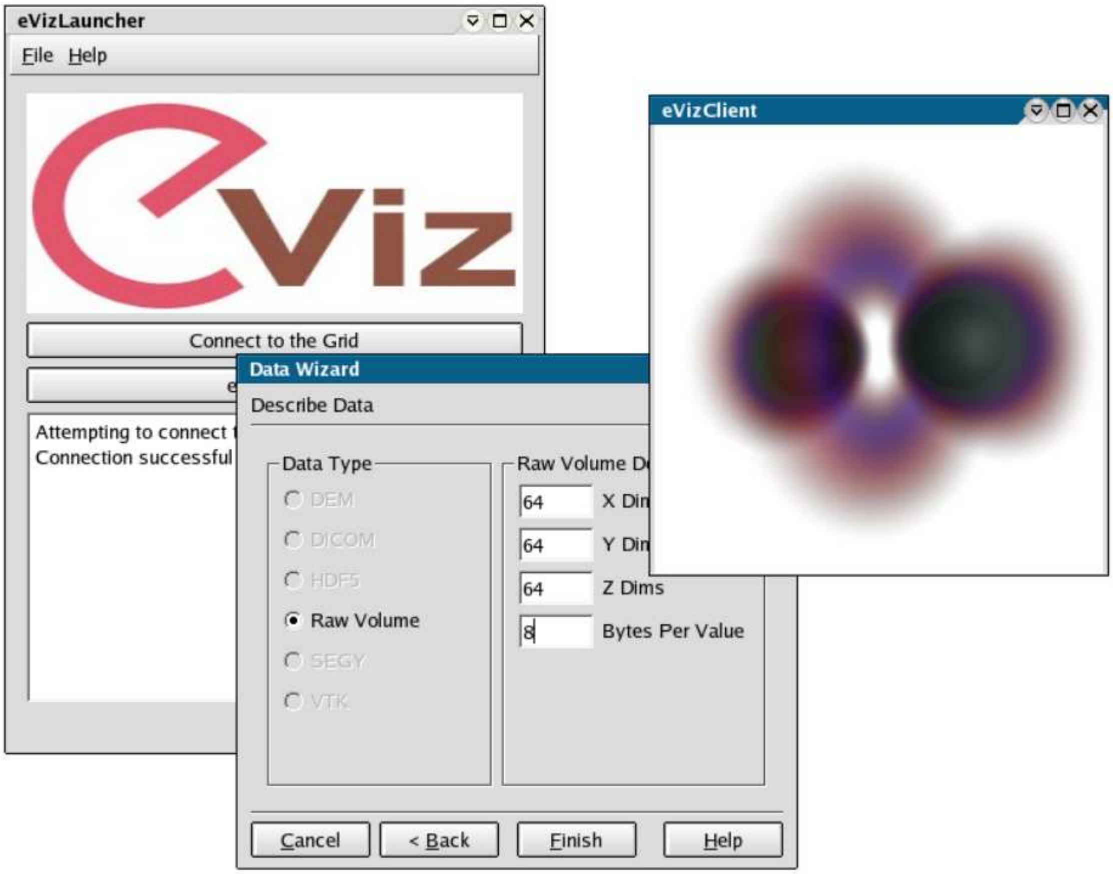


Figure 6.2: The three components of the e-Viz framework.

6.3.1 Client

e-Viz provides a basic client for both setting up and controlling the visualization pipeline and for viewing the output of the remote visualization. It also provides an API that allows user's to develop their own client applications which can utilize the e-Viz resources. A *Launcher* application provides the user with an entry point into the e-Viz system as shown in Figure 6.3. It provides a wizard allowing the user to specify the source data for the visualization and their desired output. gSOAP [109] is used to connect to the broker machine using *web services* to establish what hardware and software resources are available, and to use the knowledge-base to derive an appropriate visualization pipeline for the specific job. The *Launcher* then connects to the *Grid* exchanging certificates and establishing the connection to the remote HPV resources. *The Java Commodity Grid Kit* (CoG) is used connect to the Grid negating the dependency for the user to have the *Globus* client installed on their local workstation.



Attempting to connect t
Connection successful

Data Type
○ DEM
○ DICOM
○ HDF5
● Raw Volume
○ SEGY
○ VTK

Raw Volume D
64 X Dim
64 Y Dim
64 Z Dims
8 Bytes Per Value

Figure 6.3: The standard e-Viz client providing a Launcher application with a visualization wizard, and a remote Visualization viewer. Here a volume visualization of a hydrogen molecule is being rendered.

The client user interface also contains a viewer for the remote visualization output. The viewer has the capability to connect to multiple visualization servers allowing dynamic switching between them. This allows a seamless migration between visualization servers when the system adapts to use different resources as more appropriate visualization servers become available. It also provides redundancy in mission critical applications where the pipeline can be split to provide a duplicate output stream that the views can automatically switch to should the primary visualization resources fail.

6.3.2 Server

Each server must have the *Grid middleware* installed to become operational, to allow the client to connect and run the visualization task. It is also expected that each server will have at least one visualization application installed for it to become useful. Each visualization application to be used within the e-Viz framework needs to be extended to ended to use both the gViz computational steering library [146] and the e-Viz remote rendering libraries. These libraries both form a wrapper providing an abstraction from the specific application data into a standard description, which can therefore be interpreted in the same way by different visualization software.

6.3.3 Broker

The broker uses a knowledge base to store the status of the available servers and inventory what resources they are capable of providing. By storing the system knowledge-base it is able to make decisions about which hardware and software combinations to use and how to configure the visualization pipeline appropriately. The *web service container* is implemented using WSRF::Lite [166], in which a separate instance is created for each e-Viz session allowing communication with the e-Viz client. The client can interact with the Broker by the use of gSOAP calls, which will tell the Client which visualization servers to connect to.

6.3.4 Adaptive Codec Selection

Pixel values calculated on the graphics card in each Visualization server are encoded and transmitted over the Grid to the client where it is decoded and presented in real-time to the user. In order to meet the demands of real-time visualization delivery over congested networks, e-Viz uses different methods for encoding the image stream at a cost of increased computational load and degraded image quality.

Codecs such as Colour Cell Compression (CCC), JPEG, PNG and Run Length Encoding (RLE) are implemented in e-Viz. The e-Viz library contains data collection routines which gather the following data on each codec:

- Compression time.
- Decompression time.
- Compression ratio.
- Processing power.
- Network bandwidth.

This information is then stored in the knowledge-base to assist with future decision making.

6.4 Abstract Visualization Description Language

To enable the visualization software to be interchangeable a level of abstraction must be applied to each package allowing a single *Abstract Visualization Description Language* to be compatible with each. The e-Viz system takes advantage of a visualization description language, skML [167], which was developed during the gViz project [146]. The skML description for each visualization task contains information about each of the functional components required for a visualization process. This information includes where each process exists, how the processes are interconnected and details about how each process can be accessed.

By its abstract nature the skML language allows the description of each visualization task to be translated between visualization software. For example an IRIS Explorer network can be simply translated into its OpenDX equivalent. Each visualization package should therefore produce an identical image when attempting to implement the same abstract skML pipeline - see Figure 6.4.

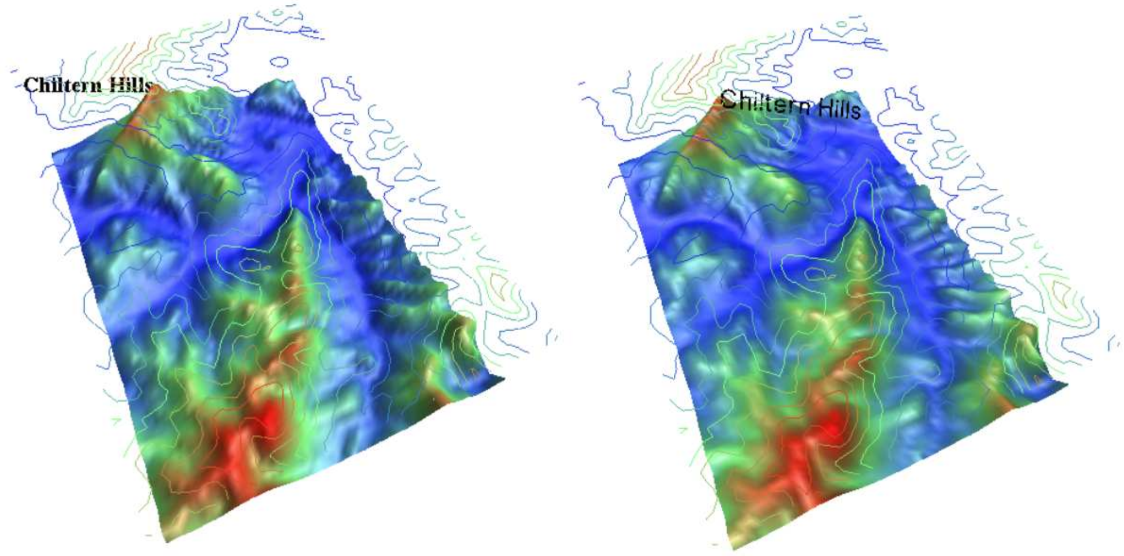


Figure 6.4: The Abstract Visualization Description Language allows a single visualization task to be rendered using different Visualisation applications - in this example AVS explorer (left) and VTK (right).

6.5 Towards Autonomic Computing

Although the e-Viz framework currently can only be classed as an *adaptive* framework many of the feature do tend towards *Autonomic Computing*:

6.5.1 Self-configuration

e-Viz follows the principles of the gViz [146] reference model, as shown in Chapter 2, where the user is provided with a number of suitable options for rendering their visualization task, specified at the conceptual level. The process of specifying the task at the logical and physical levels is left to the e-Viz system. This means that the user only has to choose the desired type of visualization and not have to understand how or where it is implemented.

6.5.2 Self-healing

In an attempt to provide self-healing to visualization tasks, the e-Viz system regularly monitors the status of each visualization component. Should a component fail to respond then the system will use the forward this information to the associated pipeline, which will either start a new service or switch to one that is already running. The method proposed by Roard [67] is used to attempt to restart failed services to ensure that the visualization quality is not degraded and happens completely transparently to the user.

6.5.3 Self-optimization

The e-Viz framework allows for self-optimization in two ways. Firstly SimuVis [148] is used to provide simulations of each of the possible pipeline configurations. This returns statistics which can be used whilst the simulation is running to dynamically choose better pipeline configurations. Secondly the use of agents within the system generate statistics for each component of the pipeline. If any of the agents return lower performance statistics than expected then the system switch to faster services.

6.5.4 Self-protection

e-Viz attempts to satisfy the self-protection requirement through redundancy. If the visualization task is needed for a mission critical application, then multiple services will be run in parallel. The redundancy strategy within the system will then switch between services, should one service fail.

6.6 Advantages of using e-Viz

The e-Viz framework is designed to allow application scientists to be able to utilize Grid enabled HPV resources to visualize their data, without needing a prior knowledge of either the Grid or visualization techniques. BART differs from these traditional tasks, by utilizing the e-Viz broker to identify remote resources not only that supply visualization services but also computational services to perform the feature point extraction and pose estimation. BART also uses the e-Viz API to develop its own client software, which not only displays the visualization, but composites it into the video stream of the users view.

6.7 Demonstrating the e-Viz framework

To be able to use the e-Viz framework as part of the BART application, it was necessary to write our a custom client to the e-Viz system. First a proof of concept client for e-Viz using the ARToolkit [89] was developed to interact with a simple visualization task as shown in Figure 6.5.

The e-Viz framework follows the same notation as VTK, using the *Eye position*, *Look at point* and *up Vector* points to represent the camera position in our remote visualization. The *Eye position* represents the actual position of the camera, the

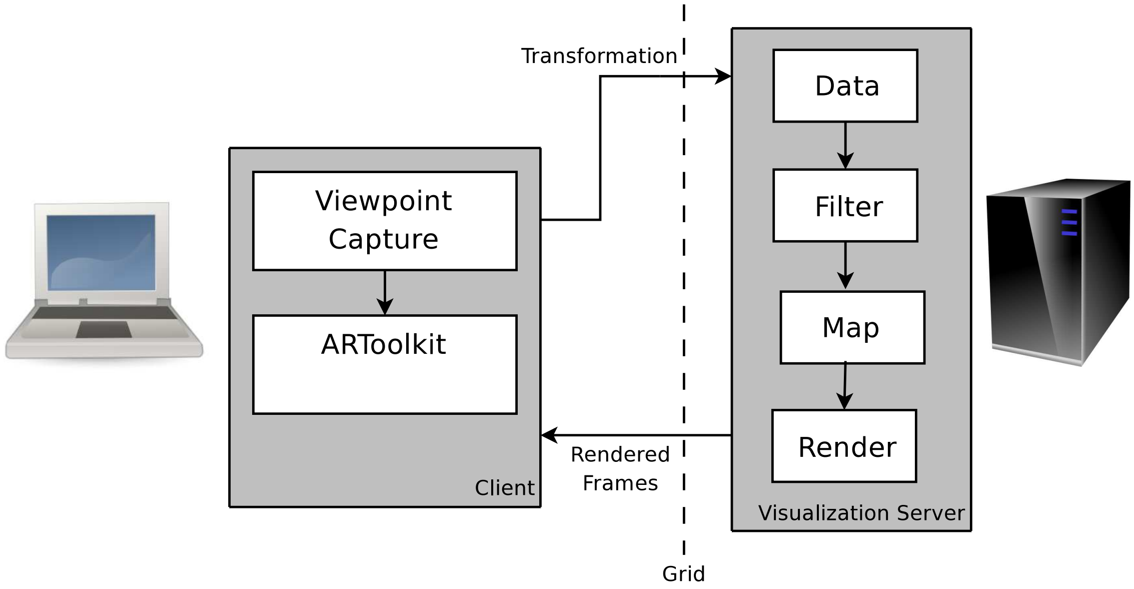
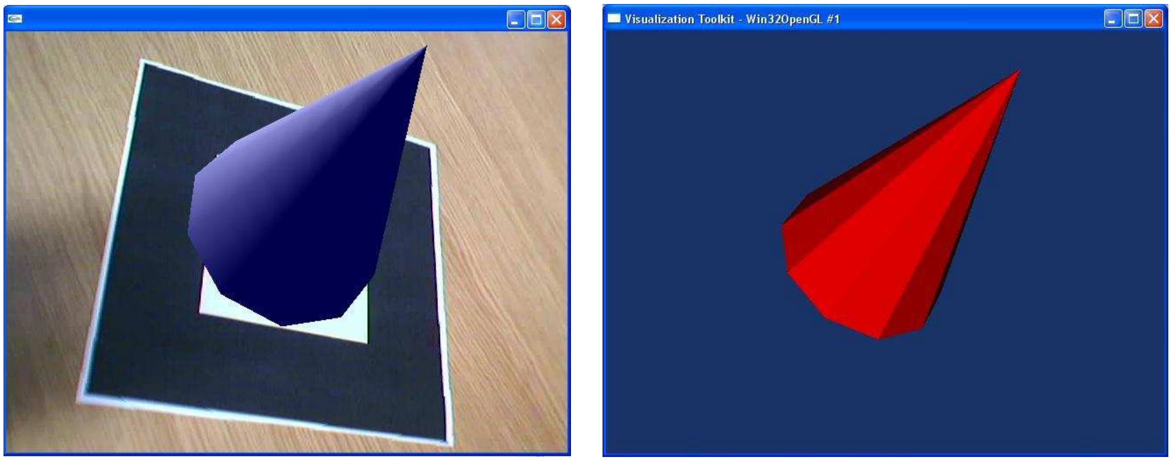


Figure 6.5: The e-Viz demonstrator using the ARToolkit to steer a simple visualization task.

Look at point represents the point at which the camera is looking and the *up Vector* represents the vector that tells us where the top of the camera is and therefore the top of the image produced by the camera. Figure 6.6 shows both the viewpoint captured using the ARToolkit and the e-Viz remote rendering of the virtual cone. It is possible to see an accurate alignment between the locally rendered cone and the e-Viz remotely rendered cone, showing that this application has been successful.

6.8 Using e-Viz for Remote Rendering

The first Grid enabled version of the software - BART v3, uses e-Viz to render the virtual artifacts present in our AR view as shown in Figure 6.7. The user's viewpoint is captured by the local machine and the pose estimation is calculated locally. The pose transformation is used to steer the e-Viz visualization pipeline, which in



(a) (b)

Figure 6.6: Using e-Viz to remotely render a simple cone using VTK: (a) The simple ARToolkit based client used to steer the simulation and (b) the remotely rendered cone is displayed on the client.

the background sends the transformation information to an available visualization server. Our client then receives the rendered image and composites it locally into the user's view.

6.8.1 Rendering the Volume Dataset with e-Viz

MRI datasets of the subjects head are volume rendered to provide the cranium view provided by BART. In the example dataset shown in Figure 6.8 it is possible to see both the original dataset and the subjects cranium which has been segmented manually using itkSNAP [168]. The dataset contains 256 × 256 × 120 voxels. Each voxel is represented using a 16-bit integer value, making the total size of the dataset 15Mb in size.

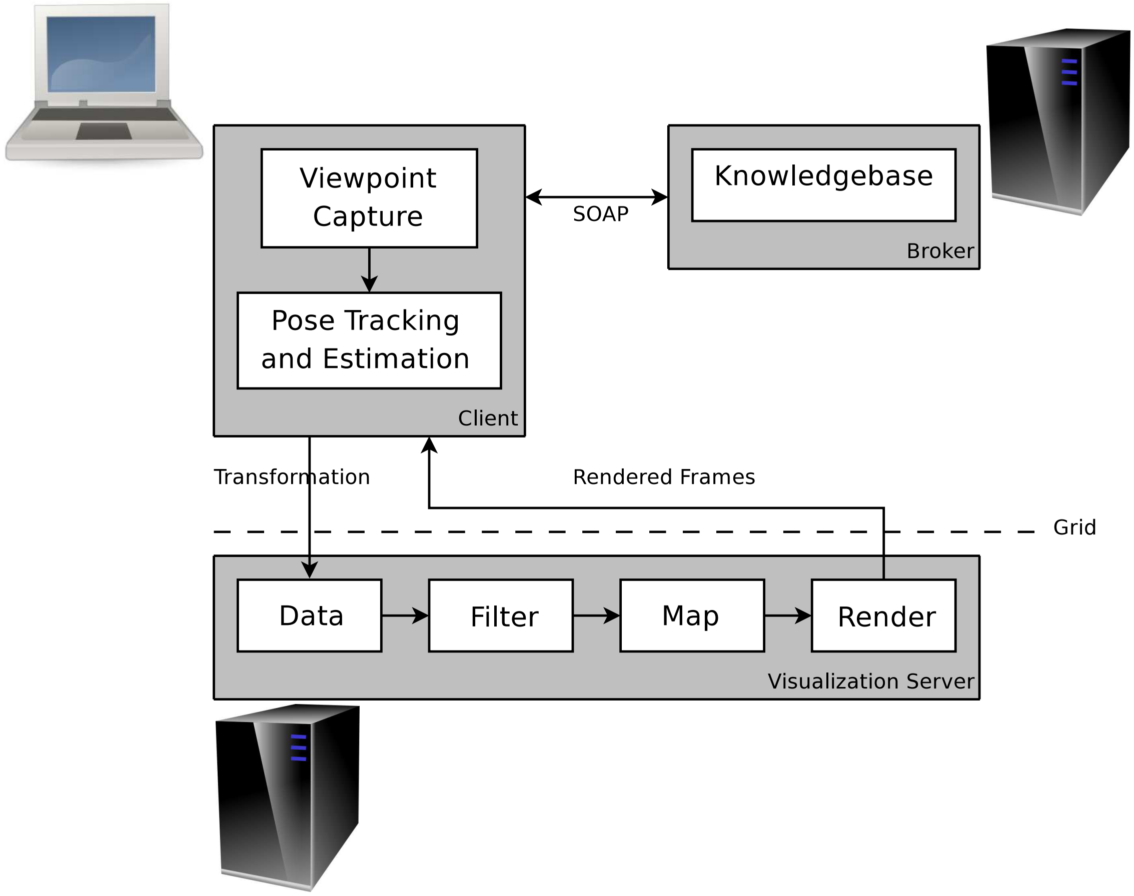


Figure 6.7: The first grid enabled version of BART, uses e-Viz to perform the remote rendering.

6.8.2 Using BART to steer the e-Viz visualization

The custom client application that was developed for this thesis, uses the e-Viz launcher to start the volume rendering job. The e-Viz broker uses its *Decision Making Module* to provide a recommendation as to which machine to start the volume rendering on and how to configure the visualization pipeline. Should no suitable resources be available then e-Viz has the capabilities to start the job on a less suitable server and dynamically switch servers once the better resources become

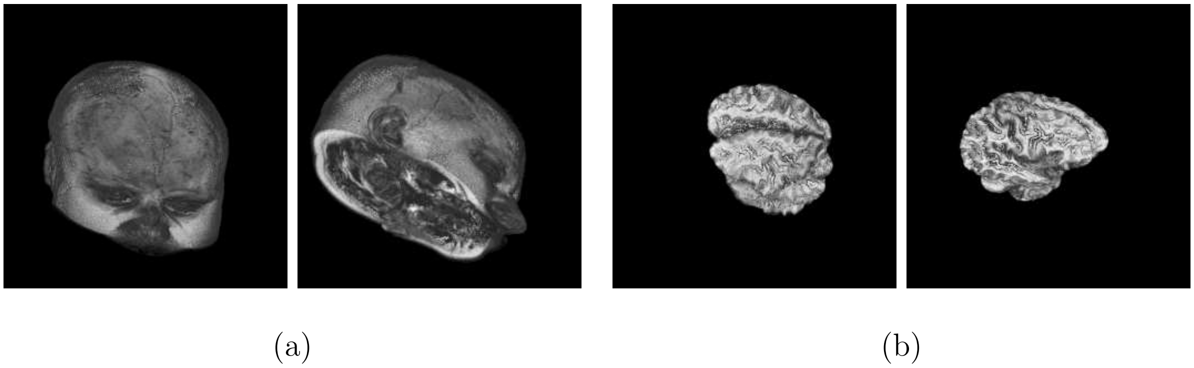


Figure 6.8: A typical MRI dataset used by BART (a) and the cranium segmented from the dataset (b).

available, in order to deliver the remote visualization as soon as possible. Many Grid based resources utilize scheduling or jobs, which can cause user's to be put into a queue whilst they wait for their time on the machine. This would be totally inappropriate with the TMS BART application as it would not be appropriate for the clinician to be forced to wait.

Once the visualization has started the remote visualization is streamed back to our client application. e-Viz currently has no support for sending alpha values to represent transparency within the rendered graphics. Therefore a coloured background to the visualization was implemented to allow it to be identified and removed to enable us to composite the rendered graphics into the user's viewpoint. During run-time as the user's viewpoint changes, BART v3 is able to calculate the updated pose of the calibrated object and use this to steer the camera in the visualization, using the gViz steering framework.

6.9 Distributing BART as part of the remote Visualization pipeline

In order to fully take advantage of the e-Viz resources the BART v3.1 implementation moves the pose estimation module onto the e-Viz visualization pipeline as shown in Figure 6.9. In this case the e-Viz visualization is steered directly by the video stream of the user's view. e-Viz is used to distribute the pose estimation module to a suitable and available resource. The pose estimation module then steers the visualization pipeline and returns the final view back to the user after compositing the artificial rendering into the real scene.

6.9.1 BART as a *Grid* enabled module

Finally a server version of the BART v3.1 software was developed to enable it to be distributed onto a HPC resource. The e-Viz launcher is now used to find both a Visualization Server and also an additional computational server which can be used to run BART. In this case the client encodes the video stream captured from the user's viewpoint using a suitable codec and passes it to the HPC machine. The pose estimation and tracking is then calculated and passed directly to the gViz interface running on the visualization server to steer the visualization.

Calibration now also takes place on the remote resource. Before the service is started the calibration views and the user specified transformation information relating to them, are passed to the server. The server performs the training steps and returns a flag when it is ready.

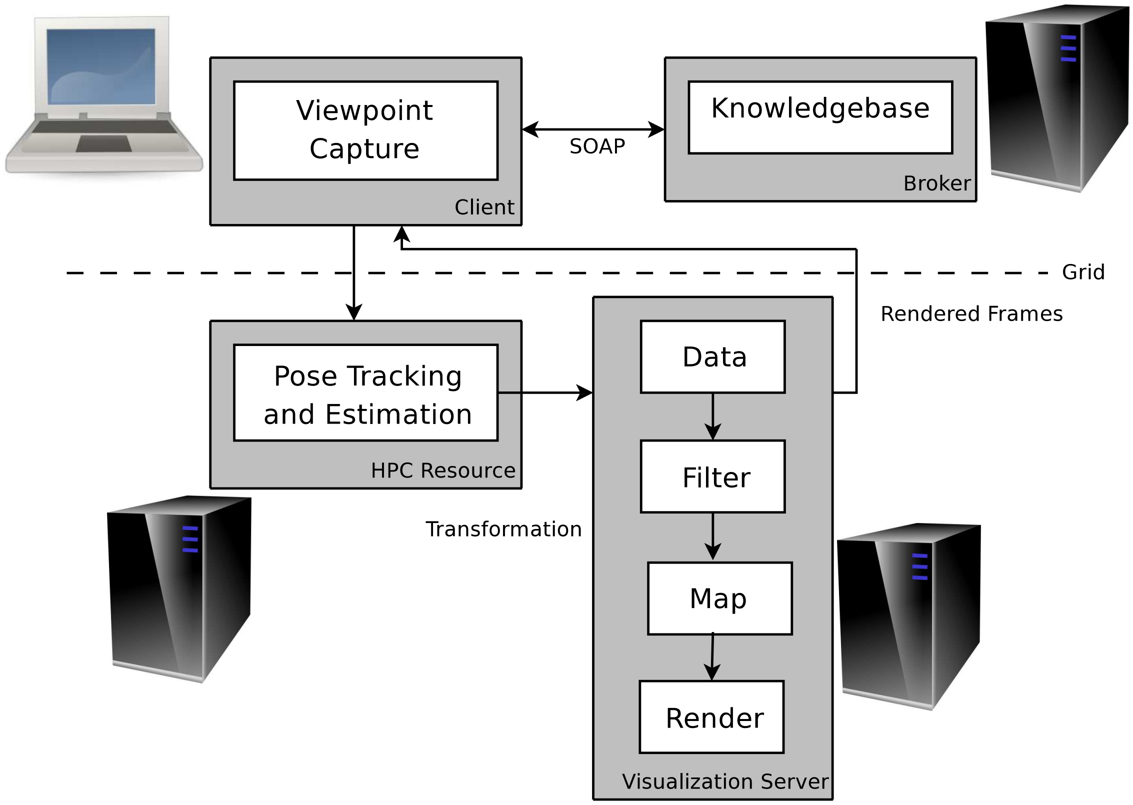


Figure 6.9: The second *Grid* enabled version of BART extends the pose tracking and estimation onto remote High Performance Computing (HPC) resources as part of the e-Viz visualization pipeline.

6.10 Results

The test video generated for Chapter 4, was run using BART v3.1. The time step was recorded by each component as it was executed, which although would add overhead time as a disk operations were involved, served to give us an indication on the proportion of time taken by each component. During this experimental setup, the following resources were used:

1. The operator's laptop, with a Sony EyeToy camera capturing the viewpoint. (1.6GHz Intel Centrino Processor, 1Gb RAM and a nVidia GeForce FX Go5200

graphics card with 32Mb).

2. The Grid based computational resource. (SGI Altix 3000, 12x Intel Itanium II Processors and 12Gb RAM)

3. The visualization server. (2x Intel Xenon Processor, 2Gb RAM and a nVidia Quatro FX graphics card with 256Mb)

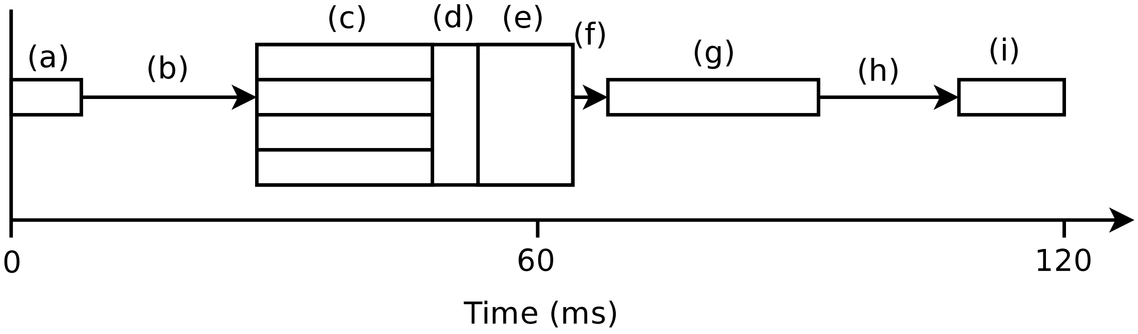


Figure 6.10: The average time taken by each component of BART v3.1. The data has been rounded for illustration purposes.

Whilst processing all 100 frames the average time to process a frame was calculated at 121.8ms, as shown in Figure 6.10. This is broken down into the nine stages as listed below, and their average time:

(a) The frame was loaded from the Operator's laptop and compressed using JPEG compression. (7.6ms)

(b) This was transferred to the Computational resource using Globus. (23.3ms)

(c) The trained cascaded Haar-classifier utilizes the multiple processors of the computational resource to find a match between the list of feature points from the viewpoint image and each of the calibration images. (18.7ms)

(d) The Harris Operator is used to generate a list of feature points from subsection of the image which has been identified by the Haar-classifier. (3.8ms)

(e) The POSIT algorithm calculates the pose for the virtual object based upon the feature points. (12.2ms)

(f) The transformation information is sent to the visualization server to steer the visualization using the e-Viz. (3.6ms)

(g) The subject's cranium is remotely rendered in VTK. (22.8ms)

(h) The rendered image is transferred back to the operator's laptop, using a similar JPEG compression, and the e-Viz framework.(17.6ms)

(i) The captured viewpoint and the rendered graphics are composited and presented to the user. (12.2ms)

When using the JPEG compression to transport the viewpoint image, it was found that the typical 640 × 480 image capture was producing data between 20Kb and 25Kb. Although the data returned from the renderer matches the resolution of the source image, it was found that the data size was typically smaller because the singular coloured background tended to compress more easily with JPEG, and therefore the time taken by (h) was always less than the time taken by (b).

Although we found we were able to process each frame in around 121.8ms, this still only provides an average frame rate of 8.2FPS, which was not enough to satisfy the conditions for an AR environment. The computing resources also were being wasted as they had to wait for each other to complete previous steps.

The solution to this was to stagger the processes as shown in Figure 6.11. BART was forced to provide a new frame to the system at the desired frame-rate that can be specified by the user. As soon as each image is received by the pipeline, it is processed. A basic management strategy was used to ensure that any old frames arriving at the client were discarded. By utilizing the resources more efficiently it was found that we could generate composited images at the same rate that they were being produced, just at the cost of being delayed by one frame. Figure 6.12 shows the time taken between compositing frames when using our test video, giving an average time of 68.0ms, which is about 15FPS. It was also found that provided the virtual artifacts were aligned with the image that they were matched to, the operator was unable to notice the slight latency.

In Figure 6.12, it is also possible to see the two computational peaks where the camera was moving between calibrated views.

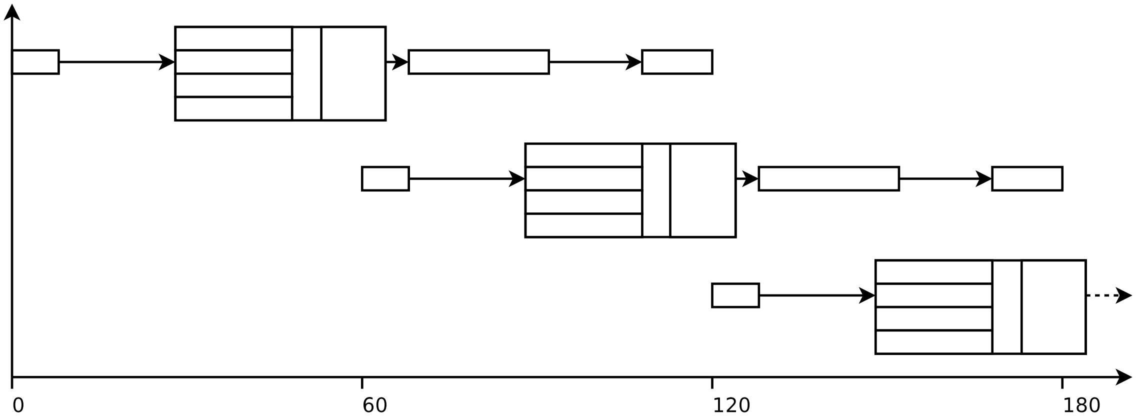


Figure 6.11: The distributed nature of the BART v3.1 system allows staggered processes to be overlaid.

Further testing was done to evaluate the robustness and scalability of the system. The e-Viz visualization servers that were available via the e-Viz broker included:

- Pentium III, 1Ghz, 1Gb RAM, Riva TNT Graphics.
- Athlon 3200, 1Gb RAM, Nvidia Graphics.

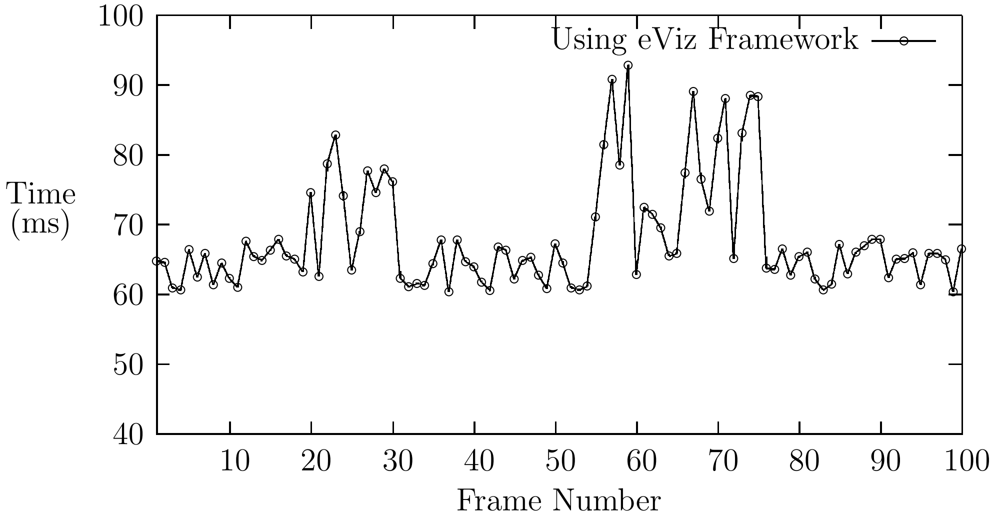


Figure 6.12: This graph shows the time between each frame of our test video that is returned to the user.

- 12CPU SGI Altix 3000, 12Gb RAM.
- 256CPU SGI Origin 3000.

Each of these services offered VTK as the software rendering tool and it was found that the visualization could be switched between each of the servers, by synchronizing the visualization state and switching the visualization stream. The data was rendered at an appropriate resolution to ensure real-time performance was maintained depending upon the capabilities of the server. When using the SGI Origin 3000 as the visualization server, which requires each job to pass through a batch queue, the e-Viz broker dynamically started the job on a lower powered but interactive machine to ensure that visualization was available quickly. As soon as the more powerful machine was available the e-Viz client switched dynamically. Testing was also done to see how the e-Viz framework handled servers failing. By manually killing the visualization tasks e-Viz dynamically switched to the next available resource, whilst the visualization was automatically restarted on the failed machine.

6.11 Conclusions

This research has shown that although although an average desktop PC does struggle with the pose estimation, using remote resources can ensure real-time performance. Provided the visualization server is appropriate for the rendering task the e-Viz framework is able to return the rendered artifact to the user at a reliable 15 FPS, where there slight latency. On congested networks the e-Viz framework, compressed the rendered images using a variety of codecs, including JPEG, PNG and run-length encoding, and transmitted them to the client. BART was therefore able to use stricter compression algorithms at a cost to the image quality to try and maintain these usable frame rates.

Chapter 7

Conclusions and Suggestions for Future work

This thesis we set out to satisfy the following hypothesis:

'It is possible to create an interactive Augmented Reality interface, using feature points for registration. It will also be capable of providing high performance visualizations by seamlessly deploying processing power from a remote, possibly Grid-enabled, high performance computing resource.'.

This hypothesis was tested using a real world application called Transcranial Magnetic Stimulation to evaluate the system. This Chapter summarizes the conclusions and examines whether or not the hypothesis has been satisfied.

7.1 State of the Art

Chapter 2, explored the state of the art for AR and looked at many of the supporting technologies, ranging from immersive displays through to the visualization software and hardware required to render the high quality 3D graphics. The chapter also looked at how the computational Grid is being used to distribute visualization tasks to remote HPV resources in response to the demand for rendering larger data-sets at higher resolutions, whilst still providing interactive real-time frame rates.

In order for the AR environment to be successful and accepted by the user, three cases must satisfied:

1. The real and virtual world must be combined with an accurate alignment between the two.

2. The environment must be interactive in and real-time (i.e. more than 15 frames per second).

3. The *Computer Graphics* must be registered in 3D.

Furthermore, chapter 2 identified several approaches to AR, although the most successful of these rely upon bold markers to align the virtual artifacts with the real world. There had also been very little research done into using HPV resources to render the virtual objects within the AR scene and to use the pose estimation to steer the virtual camera in the remote simulation.

The Autonomic Computing Model from IBM was also identified, which states applications much provide four functions to be truly autonomic: Self-configuring, Self-healing, Self-optimizing and Self-protecting.

This review was used as the motivation for a State of the Art (STAR) Report presented at Eurographics 2004, in Grenoble, by the e-Viz consortium and was later refined to become a Journal publication in the Computer Graphics Forum in 2005.

7.2 Bangor AR for TMS (BART)

In order to evaluate the Hypothesis, *Bangor AR for TMS* (BART) was developed to provide a generic AR interface which can be used in real world situations. Three versions of the software were developed as steps towards finally providing a solution which was able to fully satisfy our hypothesis. These software steps are concluded below:

7.2.1 BART v1

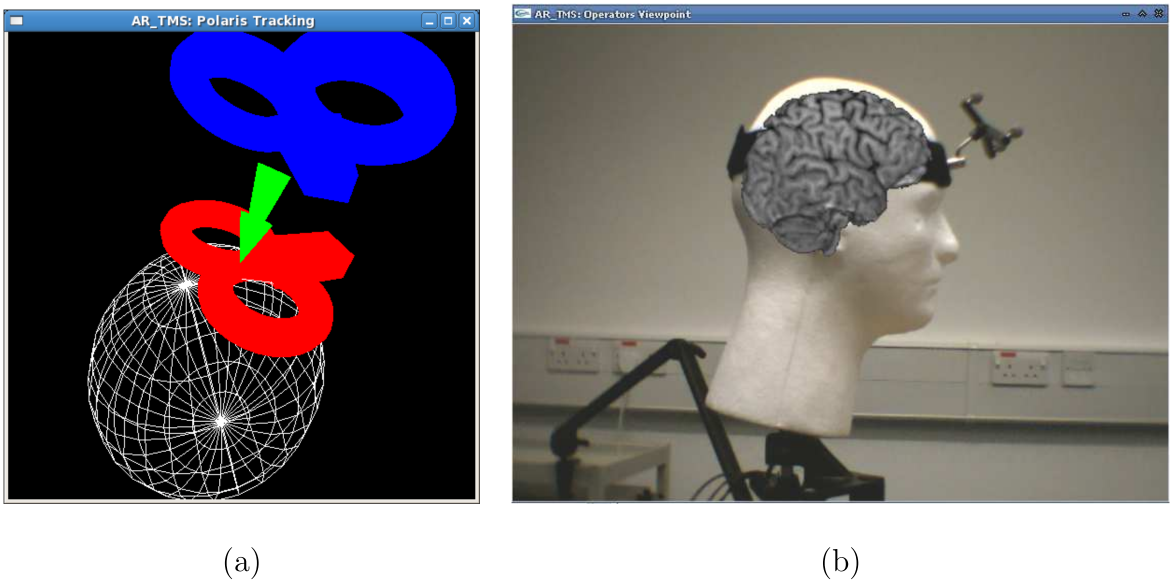
BART v1, used a *Polaris Optical Tracking System* to track the position and orientation of both the subject's head and the user's viewpoint. This gave us the scope to ensure that our software was able to calculate the translations between the user's view and the subjects head, and therefore ensure an accurate alignment of the computer graphics to the real world. Also a targeting tool was developed to allow the operator to specify points within the cranium and provide a guide arrow within the user's viewpoint to direct the movement of the tools towards the target.

Although a full validation study is outside the scope of this project, favourable comments about the ease of use of the AR interface were obtained from the Brainsight operator's in the School of Psychology. The alignment was visually acceptable, as shown in Figure 7.1, and the AR interface for positioning the coil was more natural than having to look between the subject and the results displayed on the workstation monitor (even if the workstation display is also projected onto the wall).

This was the fist time that an AR interface had been implemented for TMS, by any research group. As a result a paper was accepted for oral presentation at MMVR 2006 based upon this work.

However there are some limitations as to how well the software could perform.

The laptop computer only had basic graphics capabilities and although is capable of rendering the subject's brain at an average 15FPS, it can only render the graphics at a very low screen resolution, whilst maintaining this real time performance. It is also not possible to composite the rendered cranium into the user's viewpoint without vastly reducing the FPS and losing the realism of the AR environment. However these issues are addressed and solutions are found in subsequent versions of our software.



(a) (b)

Figure 7.1: The BART v1 software showing two of the operator's views: (a) The targeting tool helping to align the coil with a region of interest the subjects cranium and (b) a rendered view allowing the operator to see inside the subjects head.

7.2.2 BART v2

The second version of the software, BART v2, concentrated on moving away from proprietary optical tracking systems and to attempt to provide a new tracking and pose estimation using the feature points which have been extracted from the user's viewpoint. A Harris [103] corner detection algorithm was implemented to extract repeatable feature points from the operator's viewpoint. A Cascaded Haar-classifier [159] was then used to estimate the region within the viewpoint that currently contains the real world object to which we are mapping our virtual artifact. A POSIT approach [160] was then used to estimate the pose of the real world object, based upon an initial calibrated view and the comparison of feature points between the two images, as shown in Figure 7.2. By removing the need for expensive optical tracking equipment our software provides an inexpensive solution, making the procedure more accessible to training and further research.

Consequently BART v2 has been successful at providing the alignment of the virtual and real world objects, provided that optimum environmental conditions are available. However problems occur when trying to run the AR software on a local workstation. It is simply too computationally intensive and so can not keep up with real time video streams.

BART v2 implemented a novel application for combining feature point detection, with multiple Haar-classifiers for object detection. This work was presented in a poster presentation at UK e-Science Programme All Hands Meeting in 2006.

7.2.3 BART v3

In the final version of BART, the problem of addressing computational costs was explored and solved by distributing the pose estimation to a more powerful remote grid resource. BART v3, integrates the e-Viz [147] framework with High Performance Visualization (HPV) resources from remote Grid enabled facilities. Two approaches were followed to integrate BART with e-Viz. In the first version e-Viz was simply used as a remote rendering tool for the volume data-sets needed for our TMS ex-

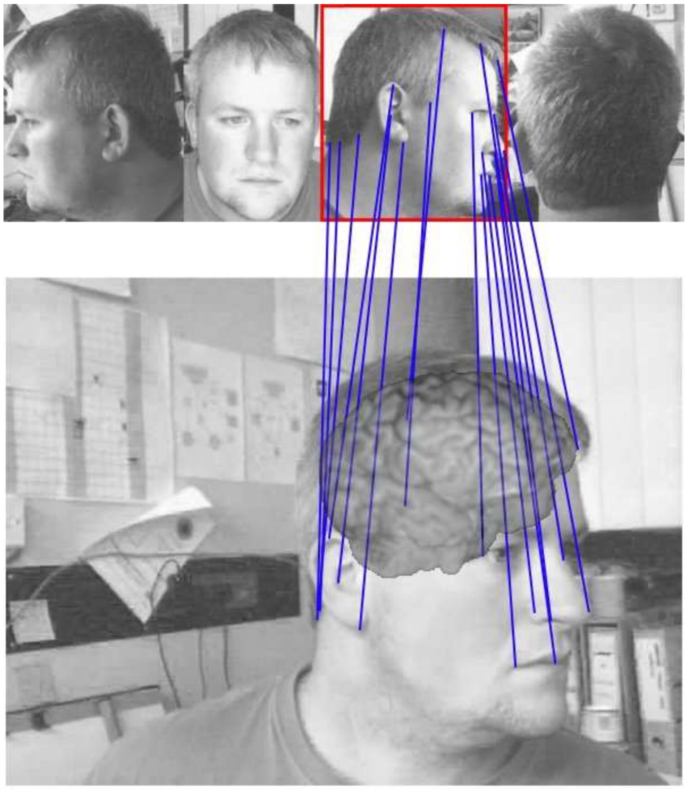


Figure 7.2: The pose tracking and estimation from BART v2.

ample. In order to solve the computational requirements of the pose tracking and estimation components, e-Viz was used to dynamically locate a suitable High Performance Computing (HPC) resource and distribute the computational load to it as part of the visualization pipeline.

This thesis demonstrates that although an average desktop PC is unable to perform our pose tracking and estimation in real-time, using remote resources can ensure real-time performance. Provided the visualization server is appropriate for the rendering task, the extended e-Viz pipeline is able to return the rendered artifact to the user at a usable frame-rate, as shown in Figure 7.3 and with only slight latency. On congested networks BART is able to compensate by using stricter compression algorithms at a cost to the image quality to try and maintain these usable frame rates.

This later work was shown in a poster that was presented at IEEE Visualization 2006 in Balitimore, as an exemplar application. Further details we provided in two papers at The Theory and Practise of Computer Graphics at Bangor in 2007, one detailing the contribution to the e-Viz framework as an exemplar application and one detailing the AR environment.

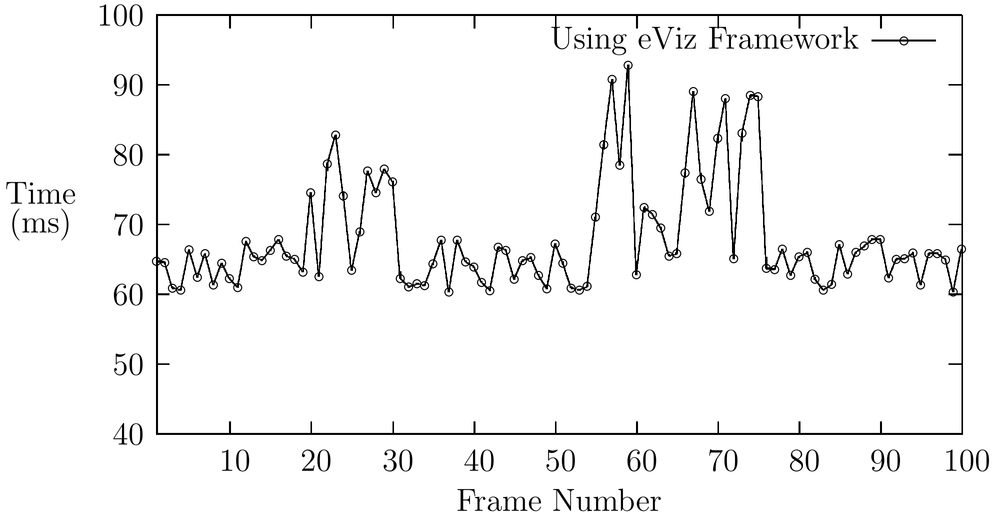


Figure 7.3: This graph shows the time between each frame of our test video that is returned to the user.

7.3 Conclusions

In order to satisfy the hypothesis the three points listed in Figure 7.4 need to be satisfied. By following the three stage software development it is clear to see that each iteration of BART provides better solutions. The final version of the software fully satisfied the hypothesis, therefore it has been proved.

Hypothesis	*BART v1*	*BART v2*	*BART v3*
Create an AR environment	Yes	Yes	Yes
Use Feature Points for registration	No	Yes	Yes
Seamlessly deploy remote HPV resources	No	No	Yes

Figure 7.4: Summary of the contribution towards evaluating the hypothesis provided by each software version of BART.

Additionally this thesis makes the following contributions to the state of the art:

- A novel AR interface to the *Transcranial Magnetic Stimulation* (TMS) using a *Polaris Optical Tracking System* has been developed. This allowed the operator to identify and align the TMS tools with the required regions of interest on the subject's cranium, whilst working in a much more natural environment. The software, *Bangor Augmented Reality for TMS* (BART) allowed the operator to see the actual rendering of the subjects cranium whilst interacting with them rather than focusing away from the subject and looking at a computer monitor.

- The software was extended to use a framework for tracking, using only a cheap webcam rather than an expensive proprietary tracking system. By extracting feature (or corner) points from the user's viewpoint the system was able to track the position and pose of the real world object that was being used for reference. A Haar-classifier technique was combined with a POSIT algorithm to track the position of the moving object and estimate its change in orientation and rotation.

- In order to use BART in real-time with large volume datasets, the e-Viz framework was successfully used to transparently allocate a remote *High Performance Visualization* (HPV) resource to remotely render the virtual artifacts.

- To satisfy the large computational requirements of the BART software it was finally distributed as part of the visualization pipeline. The e-Viz framework was used to allocate remote *High Performance Computing* (HPC) resources.

- During this research a significant contribution was made to the development and testing of the e-Viz framework, a project which was rated as tending to outstanding by the EPSRC reviewers.

7.4 Autonomic AR?

By utilizing the e-Viz framework and following its Autonomic approach to visualization, this work demonstrates the potential of Autonomic Visualization and how resource allocation could impact the way future Augmented Reality applications are implemented and how users and developers can expect to interact with them. BART goes further than the hypothesis and provides an Autonmic environment for AR, by questioning the four conditions IBM laid out for autonomic applications: [66]

7.4.1 Self-configuration

BART uses the e-Viz broker to allocate resources for the computational tasks. It also inherits the knowledgebase that the e-Viz framework provides to enable self-configuration of the visualization pipeline. By allowing the user to capture multiple views of the real-world object to track, BART also dynamically organizes the parallel cascading Haar-classifiers to perform the object tracking.

7.4.2 Self-healing

In an attempt to provide self-healing to visualization tasks, the e-Viz system regularly monitors the status of each visualization component. Should a component fail to respond then the system will use the forward this information to the associated pipeline, which will either start a new service or switch to one that is already running. Although the e-Viz framework was extended to monitor the computational resource, no provision was made for restarting the pose estimation services should they fail.

7.4.3 Self-optimization

SimuVis [148] allows the simulation of visualization pipelines and therefore can assist in the self-optimization of the visualization pipeline. There are currently no parameters within the BART application which can be changed during run-time and so provides no mechanism for self-optimization.

7.4.4 Self-protection

e-Viz attempts to satisfy the self-protection requirement through redundancy. If the visualization task is needed for a mission critical application, then multiple services will be run in parallel. The redundancy strategy within the system will then switch between services, should one service fail. Using the e-Viz frame work it would be possible to run the pose estimation service on multiple servers and although there is no provision for migration between servers, the visualization server could choose to switch pose estimator should one fail.

7.4.5 Autonomic Computing Conclusions

Although this work moves further towards producing an Autonomic solution, this has not quite been achieved. In order for BART to be fully Autonomic it must be able to make *intelligent* choices about the system configuration. Currently the e-Viz knowledgebase is fairly limited and has to give the user a list of choices rather than being able to make the decision *intelligently* by itself. Therefore the e-Viz framework has clearly achieved the Adaptive level, on the Autonomic Computing deployment model, described in Chapter 2.

However, further work would need to be done to turn BART into an autonomic solution to AR. A mechanism for restarting the remote service is needed to enable self-healing, and work still needs to be done to expose parameters from the pose estimation service, which will enable it to be optimized during run-time. Therefore BART can only be described as a Predictive system, because although it does use basic knowledge to organize the parallel cascading Haar-classifiers, it does not provide any mechanism for adaption.

7.5 Future Research Directions

7.5.1 Pose Tracking and Estimation

Future work is to concentrate on improving the efficiency and reliability of the feature point detection algorithms, ensuring that we have more accurate pose estimation between frames. Heuristics should be used to help predict the position of the virtual artifact, even if the system is unable to calculate the pose of the object, by building up a knowledge base of previous frames. There are several approaches that can be used to achieve this such as the optical flow techniques that are used by Mooser et al, to predict the movement of an object after it has been identified within the camera view [169]. Using the optical flow, it would be possible to predict where the object had moved to in the case that we had not found a correct match from the classifier. This information could also be used to simplify the pose estimation task by narrowing down the options as to the position and orientation of the virtual artifact.

Using filters to separate the tracked object from the background could help making the registration more efficient, as less incorrect feature points will be identified. This in turn would lead to a lower computational cost, as less feature points will need to be matched.

The accuracy of the registration could be improved by using multiple cameras. This would not only expand the area in which the object can be seen by the camera, but also provide a better registration should part of the object be occluded from one camera. When adding the second camera the computational load and bandwidth for the pose tracking stage would be doubled. However the two cameras could then work in competition and which ever finds the highest weighted image can be assumed to have the best view. This method would be limited by the inaccuracy of the calibration between the cameras and may experience a jump which switching cameras, if the calibration is not correct.

In order to avoid the rendered image from drifting when switching between calibration images some basic smoothing could be implemented. This would be particularly important when the camera is in the mid point between calibrations and as a result the calibration is constantly switching. It may also be possible to use a panoramic stitching techniques to create a single image from each of the calibration images which could be wrapped around a cylinder. Without the abrupt edges between images the transitions will naturally be smoother.

BART uses the Harris operator to extract feature points because it is reliable and produces repeatable feature points. It is however very computationally intensive and does take advantage of distributed HPC, to ensure real-time performance. If a more efficient algorithm still provided good enough results then it would be possible to make the system more efficient.

Since this work began in 2003, a lot of research has been done into using the Graphics Processing Unit (GPU) for processing data, rather than just graphical applications as it was designed. It would be possible to port the pose tracking and estimation software to run on a GPU, which could be fast enough to make it usable on a simple desktop computer without the need for HPC resources.

7.5.2 Autonomic AR

In order to make BART fully autonomic it would be necessary to modify the remote server to allow its status to be both queried and restarted. This would allow the e-Viz broker to properly monitor the status of the server, and if a problem has occurred to restart it. e-Viz also provides facilities for migrating services between servers, and running multiple servers in mission critical situations. These facilities would then be inherited by BART ensuring a more robust system.

In order to allow the software to self-optimize, it would also be necessary to expose each of the parameters of the cascaded haar-classifier. This would enable the e-Viz knowledgebase to provide configuration information for each task, and potentially change the running configuration whilst the system is running.

7.5.3 Conclusion

Chapter 1 stated, that in order for any AR system to be successful it needed to meet the following criteria:

- Must be easy to calibrate offline which real world objects are being tracked.
- Needs to be capable of automatically reinitializing should the alignment drift or an error occur.
- Needs to extract feature points reliably and repeatably.
- Must have a computational cost that can be satisfied in real-time.
- Must provide an accurate alignment of the virtual and real objects.
- Must work in unconstrained environments, such as changing levels of light.
- must be able to be adapted for each required application.

BART provides a very simple to use calibration tool, which allows the operator to take multiple views of an object within an environment, and then manually calibrate the virtual objects by aligning it with the environment. By always referencing the original calibration images BART constantly re-initializes therefore drift could never occur. Using the Harris Corner detector [103] provides an accurate and repeatable set of feature point which are used for reference.

The high computational cost required to perform both the pose estimation and the visualization is addressed by utilizing remote grid enabled HPC resources and the accurate alignment of the virtual and real world objects is achieved using the POSIT algorithm [160]. The generic nature of the calibration tool allows the BART software to be adapted very easily to many different applications simply by loading different data sets to represent the virtual objects.

The BART software does not however work well in unconstrained environments, and future work must be done to ensure an accurate registration in changing levels of light.

Although this thesis developed a specific AR application, it would now be straight forward to enable real-time performance for any other AR (or visualization) application that has high computational requirements. For example, the Op-3D project [144] used Grid resources to deliver volume rendered patient data to the operating room. Developers can now overlay this volume rendered data onto a video stream of the patient to give the surgeon a truly enhanced interface for hepato-pancreatic (in the Op-3D example), or any other surgery procedure. There are also many possibilities from other application domains.

Bibliography

[1] W. A. Fetter, *Computer Graphics, Aircraft Applications*, Document No. D3-424-I, Boeing Airplane Company, Wichita Division, 1961.

[2] I. E. Sutherland, *The Ultimate Display*, Proceedings of IFIP, vol 2, pp. 506-508, 1965.

[3] Kelly, K., A. Heilbrun and B. Stacks, *Virtual Reality; an Interview with Jaron Lanier*, Whole Earth Review, 1989, no. 64, pp. 108(12)

[4] Thomas P. Caudell and David W. Mizell. *Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes*, HICSS, pages 659-669, 1992.

[5] Ronald T. Azuma, *A Survey of Augmented Reality*, Presence: Teleoperators: Virtual Environments, 6,4, pp. 355–85, 1997, Hughes Research Laboratories, 3011 Malibu Canyon Road, MS RL96, Malibu, CA 90265

[6] S. Smith and T. Marsh and D. Duke and P. Wright, *Drowning in immersion.*, In Proceedings of UK-VRSIG'98. UK Virtual Reality Special Interest Group, 1998.

[7] Pedersen M L, Buckley D, Davis H, *Are the same effects found for stereogram stimuli presented either with shutter-glasses or as red/green anaglyphs?* Perception 29 ECVP Abstract Supplement, 2000

[8] *StereoGraphics*, www.stereographics.com

[9] M. Mokhtari, F. Lemieux, F. Bernier, D. Ouellet, R. Drouin, D. Laurendeau and A. Branzan-Albu, *Virtual Environment and Sensori-Motor Activities: Visualization*, Journal of WSCG, Vol.12, No.1-3, ISSN 1213-6972, WSCG2004, February 2-6, 2003, Plzen, Czech Republic.

[10] *Inition*, http://www.inition.com/

[11] *SenseGraphics*, http://www.sensegraphics.com/

[12] Gilson S J, Fitzgibbon A W, Glennerster A, *Head-mounted display calibration using camera calibration techniques* Perception 36 ECVP Abstract Supplement, 2007

[13] Roy, T.M., Cruz-Neira, C., and DeFanti, T.A. *Cosmic Worm in the CAVE: Steering a High Performance Computing Application from a Virtual Environment.*, PRESENCE: Teleoperators and Virtual Environments 4. no. 2, 1995

[14] *Fakespace systems*, http://www.fakespace.com/

[15] C. Pinhanez, *Augmenting Reality with Projected Interactive Displays*, In Proc. of International Symposium on Virtual and Augmented Architecture (VAA '01), Dublin, Ireland, 2001.

[16] David Larose, *A Fast, Affordable System for Augmented Reality*, tech. report CMU-RI-TR-98-21, Robotics Institute, Carnegie Mellon University, April, 1998.

[17] D. Schmalstieg and A. Fuhrmann and G. Hesina and Z. ari and L. Encarnac and a Gervautz and W. Purgathofer, *The Studierstube Augmented Reality Project*, 2000, citeseer.nj.nec.com/schmalstieg00studierstube.html

[18] Jun Rekimoto and Masanori Saitoh, *Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments*, In Proc. ACM CHI '99, Pittsburgh, PA, May 15–20 1999. ACM Press.

[19] W. E. L. Grimson and T. Lozano-Prez and W. M. Wells and G. J. Ettinger and S. J. White and R. Kikinis, *An Automatic Registration Method for Frameless Stereotaxy, Image Guided Surgery, and Enhanced Reality Visualization*, In Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition, Seattle, WA., june 1994. IEEE

[20] *The EG2003 Medical Prize*, http://www.informatics.bangor.ac.uk/ nigel/eg2003-mp/, Eurographics Association, 2004.

[21] S. Browne, J. Dongarra, G. Fox, K. Hawick and T. Rowan, *Software Reuse in High Performance Computing*, 7th Annual Workshop on Institutionalizing Software Reuse (WISR), St Charles, Illinois, August 28-30, 1995.

[22] V. Dvorak and L. Markovic, *High Performance Computing in JAVA - Fact or Fiction?*, Brno (Czech Republic), FIMU-RS-2000-11, 2000.

[23] Flynn, M., *Some Computer Organizations and Their Effectiveness*, IEEE Transactions on Computers, Vol. C-21, Sep. 1972, pp. 948–960.

[24] Kevin Burrage, *Parallel methods for initial value problems*, Applied Numerical Mathematics: Transactions of IMACS, 11,1–3, pp. 5–25, 1993

[25] R. Buyya and D. Abramson and J. Giddy, *A Case for Economy Grid Architecture for Service-Oriented Grid Computing*, 10th IEEE International Heterogeneous Computing Workshop (HCW 2001), In conjunction with IPDPS 2001, San Francisco, California, USA, April 2001.

[26] *SETI@home: Search for Extraterrestrial Intelligence at home*, December, 2003, http://setiathome.ssl.berkeley.edu/, University of California, Berkeley

[27] Mark Riding, Jason Wood, Ken Brodlie, John Brooke, Min Chen, David Chisnall, Chris Hughes, Nigel W. John, Mark W. Jones, Nicolas Roard, *e-Viz: Towards an Integrated Framework for High Performance Visualization*, Proceedings of the UK e-Science All Hands Meeting 2005, EPSRC, ISBN 1-904425-53-4, pp1026-1032

[28] C. Johnson and C. Hansen, *Visualization Handbook*, Academic Press, Inc., 2004.

[29] Uehara K., Scourse J. D., Horsburgh K. J., Lambeck K., Purcell A. P., 2006, *Tidal Evolution of the northwest European shelf seas from the Last Glacial Maximum to the present.*, Journal of Geophysical Research-Oceans, 111 (C9) C09025

[30] Lambeck K. *Glaciation and sea-level change for Ireland and the Irish Sea since Late Devensian / Midlandian time.*, Journal of the Geological Society, 153, 853-872, 1996

[31] Craig Upson and Thomas Faulhaber Jr. and David Kamins and David H. Laidlaw and David Schlegel and Jeffrey Vroom and Robert Gurwitz and Andries van Dam, *The Application Visualization System: A Computational Environment for Scientific Visualization*, Computer Graphics and Applications, 9,4, July, pp. 30-42, 1989

[32] Haber, R. B. and McNabb, D. A. (1990) *Visualization Idioms : a conceptual model for Scientific Visualization Systems*, in Neilson, G. M. and Shriver, B. D. (eds.) Visualization in Scientific Computing, 74 93.

[33] Mel Slater and Vasilis Linakis and Martin Usoh and Rob Kooper, *Immersion, Presence, and Performance in Virtual Environments: An Experiment with Tri-Dimensional Chess*, In M G (ed.), ed., ACM Virtual Reality Software and Technology (VRST), pp. 163–172, (July 1996). ISBN: 0-89791-825-8

[34] R.A. Davies, N.W. John, J.N. MacDonald, K.H. Hughes, *Visualization of Molecular Quantum Dynamics - A Molecular Visualization Tool with Integrated Web3D and Haptics*. In Proceedings Tenth Web3D Symposium, ACM Press, March 2005, pp143-150

[35] V. Popescu, G. Burdea and M. Bouzit,*Virtual Reality Simulation Modeling for a Haptic Glove*, In Proceedings of the Computer Animation,IEEE Computer Society, Washington, DC, 195, pp.195–9, 1999.

[36] B. Wilkinson and M. Allenfor, *Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers*, 2002, Prentice Hall

[37] U. Vishkin, *New-Old Grand Challenge: Parallel Computing*, 2002, University of Maryland

[38] *Grand Challenge Applications* , http://www-fp.mcs.anl.gov/grand-challenges/, Argonne National Laboratory

[39] *Expo Science - Industry Spacetime Wrinkles*, http://archive.ncsa.uiuc.edu, National Center for Supercomputing Applications (NCSA)

[40] E. Gallopoulos, E. N. Houstis and J. R. Rice, *Computer as Thinker/Doer: Problem-Solving Environments for Computational Science*, IEEE Computional Science and Engineering, 1,2, Summer, pp. 11-23, 1994

[41] D. W. Walker, *The Grid, Virtual Organizations and Problem-Solving Environments*, 2001, pp.445–448, IEEE International Conference on Cluster Computing, IEEE Computer Society, Cardiff University

[42] Mark Walkley, Jason Wood and Ken Brodlie, *A Distributed Co-operative Problem Solving Environment*, Computational Science - ICCS, Springer,LNCS 2329, pp. 853-861, 2002

[43] Cameron, G., *Modular visualization environments: past, present, and future*, SIGGRAPH Comput. Graph. 29, 2 (May. 1995), 3-4.

[44] Gabrielle Allen and Werner Benger and Thomas Dramlitsch and Tom Goodale and Hans-Christian Hege and Gerd Lanfermann and Andre Merzky and Thomas Radke and Edward Seidel and John Shalf, *Cactus Tools for Grid Applications*, Cluster Computing, 4,3, pp. 179-188, 2001

[45] S. G. Parker and M. Miller and C. D. Hansen and C. R. Johnson, *An integrated problem solving environment: the SCIRun computational steering sys-*

tem, 1998, 31st Hawaii International Conference on System Sciences (HICSS-31)

[46] *The Visualization Toolkit*, http://www.vtk.org/

[47] William J. Schroder and Kenneth M. Martin and Lisa S. Avila, *VTK User, s Guide - VTK File Formats*, 2000, Section on VTK File Formats published independently, http://public.kitware.com/VTK/pdf/file-formats.pdf, 14, Kitware Inc.

[48] Jeremy Walton, *NAG's IRIS Explorer*, In Visualization Handbook (Christopher R. Johnson and Charles D. Hansen, eds.), Academic Press, 2003.

[49] David Thompson, Jeff Braun and Ray Ford, *OpenDX: Paths to Visualization*, VIS, Inc. and Visualization and Imagery Solutions

[50] D.Rantzau, K.Frank, U.Lang, D.Rainer, and U.Wossner. *Covise in the cube: An environment for analyzing large and complex simulation data.*, In Proceedings of the 2nd Workshop on Immersive Projection Technology, 1998.

[51] *Technology Overview: Visualizing and Managing Core Sample Analysis with amira Software*, Mercury Computer Systems Inc, Technical report.

[52] Mario Valle, *STM3: a chemistry visualization platform*, Zeitschrift fr Kristallographie, vol. 220, no. 5-6, pp. 585-588, 2005

[53] Randall Bramley and Kenneth Chiu and Shridhar Diwan and Dennis Gannon and Madhusudhan Govindaraju and Nirmal Mukhi and Benjamin Temko and Madhuri Yechuri, *A Component based Services Architecture for Building Distributed Applications*, Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing, Pittsburgh, PA, Aug. 2000

[54] Steve Larkin and Andrew J. Grant and W. T. Hewitt, *Libraries to support distribution and processing of visualization data sets*, Future Generation Computer Systems, 12,5, pp. 431–440, 1997

[55] J. M. Brooke, P.V. Coveney, J. Harting, S. Jha, S. M. Pickles, R. L. Pinning and A. R. Porter, *Computational Steering in RealityGrid*, Proceedings of the UK e-Science All Hands Meeting, September 2-4, 2003

[56] S. G. Parker and D. M. Weinstein and C. R. Johnson, *The SCIRun computational steering software system*, 1997, Modern Software Tools in Scientific Computing, E. Arge and A. M. Bruaset and H. P. Langtangen, Birkhauser Press

[57] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T Maquire, T. Sandholm, D. Snelling and P. Vanderbilt, *OPEN Grid Services Infrastructure (OGSI) Version 1.0*, June, 2003, Global Grid Forum

[58] S. Larkin and A. Grant and W. Hewitt, *A Data Decomposition Tool for Writing Parallel Modules in Visualization Systems*, Proceedings of Eurographics UK '96, March 1996.

[59] S. Larkin and A. J. Grant and W. T. Hewitt, *Vipar, Libraries to Support Distribution and Processing of Visualization Datasets*, Lecture Notes in Computer Science

[60] Renato B. Pajarola, *Large Scale Terrain Visualization Using The Restricted Quadtree Triangulation*, pp. 19–26, 1998, IEEE Visualization 1998

[61] Dani Lischinski and Ari Rappoport, *Image-Based Rendering for Non-Diffuse Synthetic Scenes*, In Rendering Techniques '98, pages 301–314, June 1998.

[62] Douglass Davis and T. Y. Jiang and William Ribarsky and Nickolas Faust, *Intent, Perception, and Out-of-Core Visualization Applied to Terrain*, pp. 455–458, IEEE Visualization 1998

[63] R. Ingram and S. Benford and J. Bowers, *Building Virtual Cities: applying urban planning principles to the design of virtual environments*, Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST96), July, pp83-91

[64] W. Broll, *Interacting in Distributed Collaborative VE*, Proc. of VRAIS, 95, 1995

[65] *IBM Autonomic Computing*, http://www.research.ibm.com/autonomic/

[66] K Brodlie, J Brooke, M Chen, D Chisnall, A Fewings, **C Hughes**, N.W. John, M Jones, M Riding, and N Roard, *Visual Supercomputing - Technologies, Applications and Challenges*, Computer Graphics Forum, Number 24, Issue 2, 2005, pp. 217–245

[67] Nicolas Roard and Mark W. Jones, *Agent Based Visualization and Strategies*, In Proc. WSCG 2006, ISBN 80-86943-03-08, Pilzen, Czech Republic, 63-70, 2006

[68] M. Bajura, H. Fuchs and R. Ohbuchi, *Merging Virtual Reality with the Real World: Seeing Ultrasound Imagery within the Patient*, Computer Graphics, Proceedings of Siggraph, pp. 203–210, 1992.

[69] B. Itkowitz, J. Handley and W. Zhu, *The OpenHaptics Toolkit: A Library for Adding 3D Touch, Navigation and Haptics to Graphics Applications*, In Proceedings of the First Joint Eurohaptics Conference and Symposium on

Haptic interfaces For Virtual Environment and Teleoperator Systems, IEEE Computer Society, pp. 590–591, 2005.

[70] F. Conti, F. Barbagli, D. Morris and C. Sewell, *CHAI: An Open-Source Library for the Rapid Development of Haptic Scenes*, Demo paper presented at IEEE World Haptics, Pisa, Italy, March 2005.

[71] *H3D*, http://www.h3dapi.org/

[72] J. Allard, S. Cotin, F. Faure, P. Bensoussan, F. Poyer, C. Duriez, H. Delingette and L. Grisoni, *SOFA - an Open Source Framework for Medical Simulation*, Medecine Meets Virtual Reality, pp. 13-18, 2007

[73] S.D. Laycock and A. M. Day, *Recent Developments and Applications of Haptic Devices*, Computer Graphics Forum, vol. 22, no. 2, pp. 117–132, 2003.

[74] R. Ott, M. Gutierrez, D. Thalmann and F. Vexo, *Improving User Comfort in Haptic Virtual Environments through Gravity Compensation*, In Proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic interfaces For Virtual Environment and Teleoperator Systems, IEEE Computer Society, Washington, DC, pp. 401–409, 2005.

[75] F. H. Raab, E. B. Blood, T. O. Steiner and H. R. Jones, *Magnetic position and orientation tracking system*, IEEE Transactions in Aerospace Elector Systems, 15, pp. 709–718, 1979.

[76] Y. Liu, Y. Wang, D. Yan and Y. Zhou, *DPSD algorithm for AC magnetic tracking system*, IEEE Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems, pp. 101–106, 2004.

[77] *Biomechanics Analysis: IST Integrates Polhemus LIBERTY with MotionMonitor*, http://www.polhemus.com/?page=Motion_Case_Studies_IST

[78] *Polhemus Fasttrak* http://www.polhemus.com/?page=Motion_Fastrak

[79] *NDI Aurorta* http://www.ndigital.com/medical/aurora.php

[80] *Ascension Flock of Birds* http://www.ascension-tech.com/

[81] G. F. Welch, *An Inertial/Optical Hybrid Three-Dimensional Tracking System*, Technical Report UMI Order Number: TR95-048., University of North Carolina at Chapel Hill, 1995.

[82] *Intersense InertiaCube* http://www.inition.com/

[83] *Biomedicom*, http://www.biomedicom.com/

[84] G. H. Alusi, A. C. Tan, A. D. Linney, K. Raoof and A. Wright, *Three dimensional tracking with ultrasound for augmented reality applications in skull base surgery*, In Proceedings of the First Joint Conference on Computer Vision, Virtual Reality and Robotics in Medicine and Medial Robotics and Computer-Assisted Surgery, Lecture Notes In Computer Science, vol. 1205. Springer-Verlag, London, 511-517, 1997.

[85] *NDI Polaris Optical Tracking System*, http://www.ndigital.com/medical/polarisfamily.php

[86] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller and D. Colucci, *High-Performance Wide-Area Optical Tracking: The HiBall Tracking System*, Presence: Teleoperators Virtual Environts 10, pp. 1–21, 2001.

[87] *Open Computer Vision Library*, http://www.intel.com/technology/computing /opencv/overview.htm

[88] Ivan Poupyrev, Desney S. Tan, Mark Billinghurst, Hirokazu Kato, Holger Regenbrecht and Nobuji Tetsutani, *A Survey of Augmented Reality*, IEEE Computer, March, pp. 2–9, 2002, Sony CSL, Carnegie Mellon University, University of Washington, Hiroshima City University, DaimlerChrysler AG and ATR MIC Labs

[89] Kato, H., Billinghurst, M. *Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System*, In Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99). October, San Francisco, USA.

[90] Julian Looser, Raphael Grasset, Hartmut Seichter and Mark Billinghurst, *OSGART - A pragmatic approach to MR*, ISMAR 2006 workshop

[91] Burns, D., Osfield, R., *Tutorial: Open scene graph A: introduction tutorial: Open scene graph B: examples and applications*, Virtual Reality, 2004. Proceedings. IEEE, 265- 265, March 2004. ISBN: 0-7803-8415-6

[92] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. *DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences.* Conference on User Interface Software and Technology (UIST'04), October 24-27, 2004, Sante Fe, New Mexico.

[93] *Macromedia Director*, http://www.adobe.com/products/director/

[94] R. Freeman, A. Steed, *Interactive Modelling and Tracking for Mixed and Augmented Reality*, Proceedings of ACM Virtual Reality Software and Technology, pp. 61-64, Limassol, Cyprus, November 2006

[95] V. Lepetit and P. Fua, *Keypoint Recognition using Randomized Trees*, Transactions on Pattern Analysis and Machine Intelligence, Vol. 28, Nr. 9, pp. 1465 - 1479, 2006.

[96] T. Lindeberg, *Feature detection with automatic scale selection*, International Journal of Computer Vision, 30:2, pp. 77–116, 1998.

[97] *Instant Reality*, http://www.instantreality.org/home/

[98] A. Yilmaz, O. Javed and M. Shah, *Object tracking: A survey*, ACM Comput. Surv. 38, 4, 13, 2006.

[99] A. Baumberg, *Reliable feature matching across widely separated views*, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition: pages I:1774–1781, 2000

[100] L. Kitchen and A. Rosenfield, *Gray Level Corner Detection*, Pattern Recognition Letters, pp. 95-102, 1982

[101] M.H. Antchev, M.P. Petkova, and A. Kostov, *Hysteresis Current Control of Single-Phase Shunt Active Power Filter using Frequency Limitation*, In Proceeding (539) Power and Energy Systems - 2007

[102] R. Jain, R. Kasturi and B.G. Schunck, *Machine vision*, McGraw-Hill Inc., 1995.

[103] C. Harris and M. Stephens, *A combined corner and edge detector*, Proceedings of the 4th Alvey Vision Conference: pages 147–151, 1988

[104] Alexander Neubeck and Luc Van Gool, *Efficient Non-Maximum Suppression*, The 18th International Conference on Pattern Recognition (ICPR'06)

[105] S. M. Smith and J. M. Brady, *SUSAN: New Approach to Low Level Image Processing*, International Journal of Computer Vision 23, 1, pp. 45–78, 1997.

[106] M. Trajkovic and M. Hedley, *Fast corner detection*, Image and Vision Computing 16: pp. 75–87, 1998.

[107] M. S. Petkovic and D. D. Herceg, *On the convergence of Wang-Zheng's method*, Journal Comput. Appl. Math. 91, 1, pp. 123–135, 1998.

[108] *Web Services Architecture*, W3C Working Group Note 11 February 2004, http://www.w3.org/TR/ws-arch/

[109] Robert A. van Engelen and Kyle Gallivan, *The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks*, proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), pages 128-135, May 21-24, 2002, Berlin, Germany.

[110] S. Seely, *Soap: Cross Platform Web Service Development Using XML*, Prentice Hall PTR, 2001

[111] *Apache Axis* http://ws.apache.org/axis/

[112] *Web Services Description Language (WSDL) 1.1* http://www.w3.org/TR/wsdl

[113] *Apache jUDDI* http://ws.apache.org/juddi/

[114] I. Foster and C. Kesselman, *The Globus Project: A Status Report*, pp. 4–18, 1998, Proceedings of the Heterogeneous Computing Workshop.

[115] G. Allen and K. Davis, K. N. Dolkas, N. D. Doulamis, T. Goodale, T. Kielmann, A. Merzky, J. Nabrzyski, J. Pukacki, T. Radke, M. Russell, E. Seidel, J.

Shalf and I. Taylor, *Enabling Applications on the Grid: A GridLab Overview* , International Journal of High Performance Computing Applications, Vol. 17, No. 4, 449-466, 2003.

[116] B. Plale, P. Dinda and G. von Laszewski, *Key concepts and services of a grid information service*, Proceedings of the 15th International Conference on Parallel and Distributed Computing Systems, 2002.

[117] N. Furmento and W. Lee and A. Mayer and S. Newhouse and J. Darlington, *ICENI: An Open Grid Service Architecture Implemented with JINI*, 2002

[118] C. Catlett, *Why Global Grid Forum?*, February, 2003, www.ggf.org, http://www.gridforum.org/DOCS-Presentations/GGFOverviewFeb2003.pdf, Global Grid Forum Prenentations

[119] B. Neuman and S. Rao, *Resource management for distributed parallel systems*, in Proceedings of the 2nd Internationational Symposium on High Performance Distributed Computing, pp. 316–323, Spokane, WA, USA, 1993.

[120] Arpaci, Remzi H. and Dusseau, Andrea C. and Vahdat, Amin M. and Liu, Lok T. and Anderson, Thomas E. and Patterson, David A., *The Interaction of Parallel and Sequential Workloads on a Network of Workstations*, May, pp. 267-278, 1995, Proceedings of ACM SIGMETRIC, 95 Joint International Conference on Measurement and Modeling of Computer Systems.

[121] Steve Kubica, Thomas Robey and Chris Moorman, *Data Parallel Programming with the Khoros Data Services Library*, pp. 963-968, 1998, IPPS/SPDP Workshops

[122] Steve W. Bova and Clay P. Breshears and Henry Gabb and Bob Kuhn and Bill Magro and Rudolf Eigenmann and Greg Gaertner and Stefano Salvini and Howard Scott, *Parallel Programming with Message Passing and Directives*, Computing in Science and Engineering, 3,5, pp. 22–37, 2001

[123] H. Sandhu and S. Zhou, *Cluster-Based File Replication in Large-Scale Distributed Systems*, 1-5, pp. 91-102, 1992, Newport, Rhode Island, USA, ACM SIGMETRICS and PERFORMANCE, 92nd International Conference on Measurement and Modeling of Computer Systems

[124] A. Lazar and G. Pacifici, *Control of Resources in Broadband Networks with Quality of Service Guarantees*, IEEE Communications, Vol. 29, No. 10, October 1991, pp. 66-73.

[125] Mathilde Romberg, *UNICORE: Beyond Web-based Job-Submission*, Proceedings of the 42nd Cray User Group Conference, May 22-26,2000, Noordwijk.

[126] Dietmar W. Erwin and David F. Snelling, *UNICORE: A Grid Computing Environment*, In Proceedings of the 7th international Euro-Par Conference Manchester on Parallel Processing, August 28 - 31, 2001, Lecture Notes In Computer Science, vol. 2150. Springer-Verlag, London, 825-834.

[127] Steve J. Chapin and Dimitrios Katramatos and John Karpovich and Andrew S. Grimshaw, *The Legion Resource Management System*, Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing, April 1999.

[128] *HPCx Homepgae*, http://www.hpcx.ac.uk/

[129] *HECToR*, http://www.hector.ac.uk/

[130] *NASA's Information Power Grid (IPG)*, http://www.ipg.nasa.gov, NASA

[131] K. Nowinski, B. Lesyng, M. Niezgdka and P. Bala, *Project EUROGRID*, Pioner 2001 Conference Proceedings, pp. 187–191, Poznan, 2001.

[132] *Particle Physics Data Grid*, http://www.ppdg.net/

[133] R. Pennington, *Terascale Clusters and the TeraGrid*, Invited talk, Proceedings for HPC Asia, Dec 16-19, pp. 407-413, 2002.

[134] *HealthGrid*, http://community.healthgrid.org/

[135] *Access Grid Documentation Project*, http://www.accessgrid.org/agdp/.

[136] *Access Grid*, http://www.accessgrid.org.

[137] Rhys Hawkins, Yifan Lu Darran Edmundson and Paul Warren, *Introducing VPC: A Mixed Reality Videoproducer for the Accessgrid*, The 5th Annual Access Grid Retreat, California, USA, 26-29 April, 2005

[138] Leigh, J., Dawe, G., Talandis, J., He. E., Venkataraman, S., Ge, J., Sandin, D., DeFanti, T. A., *AGAVE : Access Grid Augmented Virtual Environment*, Proc. AccessGrid Retreat, Argonne, Illinois, Jan, 2001. (Jan 16, 2001)

[139] Kranzlmller, D., Kurka, G., Heinzlreiter, P., and Volkert, J. 2002. *Optimizations in the Grid Visualization Kernel.* In Proceedings of the 16th international Parallel and Distributed Processing Symposium (April 15 - 19, 2002). IEEE Computer Society, Washington, DC, 237.

[140] Grimstead, I. J., Avis, N. J., and Walker, D. W. *Automatic Distribution of Rendering Workloads in a Grid Enabled Collaborative Visualization Environment.* In Proceedings of the 2004 ACM/IEEE Conference on Supercomputing

(November 06 - 12, 2004). Conference on High Performance Networking and Computing. IEEE Computer Society, Washington, DC, 1.

[141] Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirchner, P. D., and Klosowski, J. T. *Chromium: a stream-processing framework for interactive rendering on clusters.* In ACM Trans. Graph. 21, 3 (Jul. 2002), 693-702.

[142] A.J. Fewings, N.W. John, *Distributed Graphics Pipelines on the Grid*, IEEE Distributed Systems Online, vol. 8, no. 1, 2007, art. no. 0701-o1001.

[143] *SGI OpenGL Vizserver 3.5 Visualization and Collaboration (Application-Transparent, Remote, Interactive)*, SGI White Paper

[144] Nigel W. John, *High Performance Visualization in a Hospital Operating Theatre*, Theory and Practice of Computer Graphics (TPCG03), IEEE Computer Society, ISBN 0-7695-1942-3, pp170-175, 2003.

[145] W. Bethel, B. Tierney, J. Lee, D. Gunter, S. Lau, *Using High-Speed WANs and Network Data Caches to Enable Remote and Distributed Visualization*, Proceedings of the SC2000 Conference, November 2000, Dallas, TX

[146] K. Brodlie, D. Duce, J. Gallop, M. Sagar, J. Walton, and J. Wood, *Visualization in Grid Computing Environments*, Proceedings of IEEE Visualization 2004, edited by Holly Rushmeier, Greg Turk and Jarke J. van Wijk, pp 155-162. 2004

[147] K.W. Brodlie, J. Brooke, M. Chen, D. Chisnall, C. Hughes, N.W. John, M.W. Jones, M. Riding, N. Roard, M. Turner, J.D.Wood, *Adaptive Infrastructure for Visual Computing*, Proc. of Theory and Practice of Computer Graphics 2007, Bangor, pp. 147-156. ISBN 978-3-905673-63-0

[148] D. Chisnall and M. Chen, *The making of SimEAC*, In International Conference on Autonomic Computing, pp.301-302, 2006

[149] T. Paus, R. Jech, C. J. Thompson, R. Comeau, T. Peters, A. C. Evans, *Transcranial magnetic stimulation during positron emission tomography: a new method for studying connectivity of the human cerebral cortex.* J Neurosci. May 1 1997; 17(9):3178-3184.

[150] T. Kujirai, M. D. Caramia, J. C. Rothwell, B. L. Day, P. D. Thompson, A. Ferbert, S. Wroe, P. Asselman, and C. D. Marsden, *Corticocortical inhibition of the motor cortex*, The Journal of Physiology, 471, pp. 501–19, November 1993.

[151] Herrmann, L. L. and Ebmeier, K. P., *Factors modifying the efficacy of transcranial magnetic stimulation in the treatment of depression: a review.*, Journal of Clinical Psychiatry, 67, 1870 1876, 2006

[152] Kerry J. Ressler and Helen S. Mayberg, *Targeting abnormal neural circuits in mood and anxiety disorders: from the laboratory to the clinic*, Nature Neuroscience Review, Nature Publishing Group, 2007

[153] Herrmann, L. L. and Ebmeier, K. P. *Factors modifying the efficacy of transcranial magnetic stimulation in the treatment of depression: a review.*, Journal of Clinical Psychiatry, 67, 1870-1876, 2006

[154] *Polaris Host Specification Guide*

[155] *Video 4 Linux*, http://linux.bytesex.org/v4l2/

[156] Woo, M. and Shreiner, D., *OpenGL Programming Guide: the Official Guide to Learning Opengl, Version 1. 4. 4.*, Addison-Wesley Longman Publishing Co., Inc., 2003

[157] Andrew J. Hanson, *Visualizing Quaternions*, Morgan Kaufmann Series in Interactive 3D Technology, ISBN-10: 0120884003

[158] G. S. Fischer, E. Dyer, C. Csoma, A. Degeut and G. Fichtinger, *Validation System of MR Image Overlay and Other Needle Insertion Techniques*, Stud Health Technol Inform., 125, pp. 130–5, 2007.

[159] Wilson, P. I. and Fernandez, J. *Facial feature detection using Haar classifiers.*, Journal of Computing Sciences in Colleges, 21, 4 (Apr. 2006), 127-133.

[160] D.F. DeMenthon and L.S. Davis. *Model-Based Object Pose in 25 Lines of Code*, International Journal of Computer Vision, Vol. 15, pp. 123-141, June 1995

[161] *Linux OVCam Drivers*, http://alpha.dyndns.org/ov511/

[162] R. Lienhart and J. Maydt, *An Extended Set of Haar-like Features for Rapid Object Detection*, IEEE ICIP, vol. 1, pp. 900-903, 2002

[163] Yoav Freund and Robert E. Schapire, *A short introduction to boosting.*, Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, September, 1999.

[164] J. Gausemeier, M. Grafe, C. Matysczok, R. Radkowski, *Optical tracking stabilization using low-pass filters*, Augmented Reality Toolkit Workshop, IEEE International, pp. 16–17, 2003.

[165] M. Clothier and M. Bailey, *Overcoming Augmented Reality Tracking Difficulties in Changing Lighting Conditions*, In Proceeding Computer Graphics and Imaging, ACTA Press, 426, pp. 84–88, 2004.

[166] *WSRF:Lite*, http://www.sve.man.ac.uk/Research/AtoZ/ILCT

[167] D. Duce and M. Sagar, *skml: A Markup Language for Distributed Collaborative Visualization*, Proc. Theory and Practice of Computer Graphics 2005, Eurographics Assoc., 2005, pp. 171178.

[168] Paul Yushkevich1, Joseph Piven, Heather Cody, Sean Ho, James C. Gee1, and Guido Gerig *User-Guided Level Set Segmentation of Anatomical Structures with ITK-SNAP*, MICCAI 2005 Open-Source Mini-Conference

[169] J. Mooser, S. You and U. Neumann, *Real-time object tracking for augmented reality combining graph cuts and optical flow*, In Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 145–152, 2007.