8-27-2012

Novel haptic interface For viewing 3D images

Mauricio Gomez Aguinaga

Mauricio Gomez Aguinaga
Candidate

Electrical and Computer Engineering
Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

Dr. Pradeep Sen, Chairperson

Dr. Thomas Caudell

Dr. Joe Michael Kniss

Novel Haptic Interface For Viewing 3D Images

by

Mauricio Gomez Aguinaga

Bachelors of Science in Computer Systems Engineering

THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science in

Computer Engineering

The University of New Mexico

Albuquerque, New Mexico

July, 2012

Dedication

I want to dedicate this thesis to the three most important persons of my life.

My parents, who have always supported me all the way through this amazing journey of learning and self discovery.

And Barbara, my sister. Thank you for being supportive when I most needed.

Acknowledgments

I heartily acknowledge Dr. Pradeep Sen, my advisor and thesis chair, for all the time he dedicated on me and for encouraging me throughout the years to be a better engineer. His guidance and professional style will be present during my whole life.

I also want to thank my committee members, Dr. Thomas Caudell and Dr. Joe Kniss, for their valuable recommendations regarding this study and for lessons learned on their classes.

Gratitude is also extended to the *Consejo Nacional de Ciencia y Tecnología* (CONACYT) and to the *Consejo Mexiquense de Ciencia y Tecnología* (COMECYT), for providing me with founding throughout the degree.

To all my colleagues in the Advanced Graphics Lab: Maziar Yaesoubi, Lei Xiao, Vahid Noormofidi, Soheil Darabi, Nima Khademi and Ian Mallett and all the other people who helped me out during my graduate studies.

Novel Haptic Interface For Viewing 3D Images

by

Mauricio Gomez Aguinaga

B.S. in Computer Systems Engineering, Tecnológico de Monterrey, 2008

Abstract

In recent years there has been an explosion of devices and systems capable of displaying stereoscopic 3D images. While these systems provide an improved experience over traditional bidimensional displays they often fall short on user immersion. Usually these systems only improve depth perception by relying on the stereopsis phenomenon. We propose a system that improves the user experience and immersion by having a position dependent rendering of the scene and the ability to touch the scene.

This system uses depth maps to represent the geometry of the scene. Depth maps can be easily obtained on the rendering process or can be derived from the binocular-stereo images by calculating their horizontal disparity. This geometry is then used as an input to be rendered in a 3D display, do the haptic rendering calculations and have a position depending render of the scene.

The author presents two main contributions. First, since the haptic devices have a finite work space and limited resolution, we used what we call detail

mapping algorithms. These algorithms compress geometry information contained in a depth map, by reducing the contrast among pixels, in such a way that it can be rendered into a limited resolution display medium without losing any detail. Second, the unique combination of a depth camera as a motion capturing system, a 3D display and haptic device to enhance user experience.

While developing this system we put special attention on the cost and availability of the hardware. We decided to use only off-the-shelf, mass consumer oriented hardware so our experiments can be easily implemented and replicated. As an additional benefit the total cost of the hardware did not exceed the one thousand dollars mark making it affordable for many individuals and institutions.

Contents

Table of Figures

Chapter 1: Motivation

The sense of sight is probably the most used for humans. It is the prime input to many other complex cognitive tasks like navigation, face recognition, reading, visual semantic or even visual attention (Bouguila, Ishii and Sato 2000). Scientists and engineers, though the ages have created many mechanisms to create and capture images along with devices that allow us to better enjoy them.

Examples of this is that today we can visually enjoy images and pictures in many different mediums and formats, varying from images printed on paper, to movies on a screen or even interactive experiences rendered in real time such as videogames. Pictures are a bidimensional projection representing a scene, which can be synthetically created or captured from optical devices such as cameras with lenses. Pictures can be a very accurate depiction of reality but usually they are limited on the amount of immersion that they can provide. This happens because pictures, even if they are captured using optical systems or rendered photorealistically, do not implement the whole set of depth perception cues (like stereopsis or motion parallax) resulting in a representation that is far from reality. When users hold a picture in their hand or look at it in a normal monitor, it merely seems like a plane where the scene was projected onto it rather than a fully immersive window to the scene.

Human perception works by combining input from diverse perceptual modalities into a unitary perception of the world (Bouguila, Ishii and Sato 2000).

When some of these perceptual modalities are lost, the illusion of being inside a virtual reality begins to vanish.

We can overcome some of these problems by building visual systems that implement the missing cues. For example, depth perception through stereopsis can be easily implemented by projecting images taken from a slightly different viewpoint for each of the eyes (Wheatstone 1838). We can implement motion parallax by tracking the position of the head and displaying a position-corrected viewpoint of the scene. We can mix these two previous concepts and implement what is called Fish Tank Virtual Reality (Ware, Arthur and Booth 93). We also decided to implement some haptic interaction using the recovered scene geometry, so that users can touch the virtual elements on the scene and have tactile feedback.

Haptic feedback in recent years has proved to be an important addition to increase user immersion. It has been widely used for medical training (Coles and John 2011) and other learning experiences (Feygin, Keehner and Tendick 2002), robot teleoperation (Çavuşoluğlu, Sherman and Tendick 2002) and even for artistic expressions (Frisoli, et al. 2004).

(Bouguila, Ishii and Sato 2000) and (Ikits and Brederson 2004) created a multimodal system that combines haptic feedback, head-tracking and uses a stereoscopic display. These two systems combine coherent visual clues with other senses to increase the user level of immersion, but they costs ranged on the hundred of thousands dollars and were not desktop suitable. Nechvatal in his book *Immersive Ideals / Critical Distance* states, while analyzing the immersive consciousness, states that the more senses you include in the experience, the more user immersion will become more engrossing (Nechvatal 2009).

In this thesis we present a multimodal system that would include the haptic experience, head-tracking for a position corrected rendering and a stereoscopic display.

We build this system with three clear goals in mind:

- Our pipeline should be able to plug in with current 3D image creation techniques.
- Our implementation should have a low resource-consuming footprint so that it can be used in systems with constrained processing and memory capabilities.
- Our system should only use off-the-shelf hardware so that it can be easily replicated and implemented. This would also keep the prices low to increase its commercial viability.

We implemented a system that takes a depth map to create a virtual stereoscopic relief that can be not only can be seen from many viewpoints, but also can provide the ability to touch it, creating a more immersive experience.

We decided to encode the scene geometry in a depth map since it can be independently calculated on either the rendering process or by calculating the horizontal disparity of a binocular-stereo image. This allows the system to have a small memory and processing footprint while providing a compact, portable representation. In this way, for example we can save the depth map in the alpha channel of an image or in an independent depth map. Since the haptic devices provide a finite workspace and resolution, sometimes geometry cannot be encoded accurately. We tested a couple of detailed mapping algorithms before the depth map gets normalized so that geometry can be saved in a regular bitmap and haptic-rendered without losing major detail.

The system rendering is very fast. It has refresh rates on the order of 1 kHz running on commodity hardware, mainly because no special rendering effects must to be applied and geometry transformations are very simple.

We used the Novint Falcon (Novint Technologies 2011) as a haptic device, the Microsoft Kinect (Microsoft Corporation 2011) as motion capture device and the Nvidia 3D Vision (Nvidia 2011) and 120 Hz 3D stereo display to have a stereoscopic output. All this hardware can be currently bought in any electronics store, and its total cost does not exceed one thousand dollars.

Chapter 2: Previous work

Most of the systems that combine the different aspects introduced in this thesis are built upon specific requirements, such as surgical teleoperation or pilot training. Most of the times, not only do they lack in open documentation but also of a general approach for regular customers since they depend a lot on application specifics.

In a virtual reality environment, the sense of immersion is greatly enhanced by a multi-modal coherent input. The addition of sensory modes greatly enhances presence in the virtual world although conflicting inputs can degrade user immersion. Ivan E. Suthernland was one of the pioneers on creating multi-modal interactive computer graphics. In (Sutherland 1968) he explored the use of a head mounted display to render 3D images. Since it is known that motion parallax is more important than stereopsis for depth perception, he registered the head movement by using a robotic arm attached to users head to interactively correct the rendered image. His results were quite impressive for the time. Users successfully perceived depth by stereopsis and motion parallax when altering the position of their heads.

(Ware, Arthur and Booth 93) assembled a system which renders a perspective-corrected image using head-tracking that could include a stereoscopic display. They called it “Fish thank Virtual Reality”. This system used a stereo camera rig in order to find and follow users head position. This system produced a very different experience than the head mounted virtual reality by

Southerland, even that both of them implemented perspective correction according to the head position and a stereoscopic display. In Fish thank Virtual Reality the display functions as a window to the virtual world rather than camera into the virtual world, providing a different level of immersion. Fish thank Virtual Reality is often stereotyped as a non-immersive virtual reality.

(Ware, Arthur and Booth 93) measured the error reduction when using perspective-corrected rendering, by comparing stereoscopic with traditional rendering. They validated that when thanks to the head-tracking, motion parallax is implemented created a stronger impression of depth than stereopsis. Depth perception error got reduced from 22% to 14.7% when using stereoscopic displays, 3.2% when using head-tracking and only 1.3% when using both head-tracking and a stereoscopic display at the same time.

(Swapp, Pawar and Loscos 2005) introduced a system that combines a haptic device and a stereoscopic display acting in a co-located space in order to model interactions with objects in the virtual world. Their system consists of a three by three projected screen and an apparatus that provided haptic feedback by having a ball connected to a set of metal threads on which the system can apply the desired forces. Their results were quite impressive.

(Arsenault and Ware 2000) introduced a system in which they combined a stereoscopic display and a haptic device acting in a collocated space adding perspective correction using head-tracking. In this study motion parallax proved to be a very important depth cue to implement on multi-modal systems in order to enhance immersion. Throughout their experiments they proved an improvement by average of 9% on placement accuracy, an improvement of 12% on contact sensing and almost a 20% of improvement on both tasks.

(Bouguila, Ishii and Sato 2000) also tried to combine stereoscopic displays and haptics in a collocated space so that users can coherently feel and see when they are grasping a virtual object. For this system they used the Big SPIDAR interface (Laroussi, Cai and Sato 97). This system consists of a series of strings, holding a ring that works as a handle on a 3D space. Each of the strings is connected to a motor that provides the needed force feedback. Then they provide the stereoscopic images through a 3D projector that uses shutter glasses. Some other work like (Ernst, Banks and Bulthoff 2000) and (Wall, et al. 2002) also proved that depth perception is improved when interaction is coupled with haptic feedback.

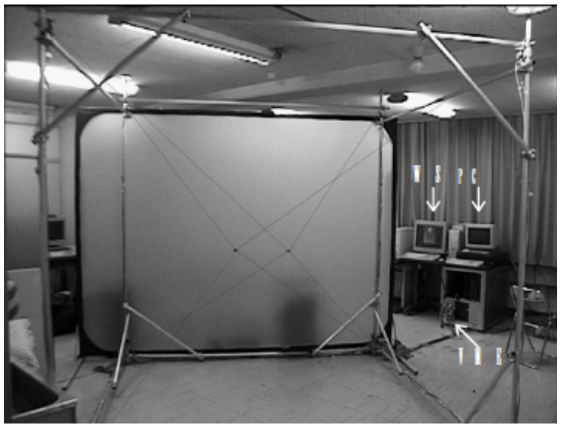


Figure 1. Apparatus of the scalable-SPIDAR (Bouguila, Ishii and Sato 2000)

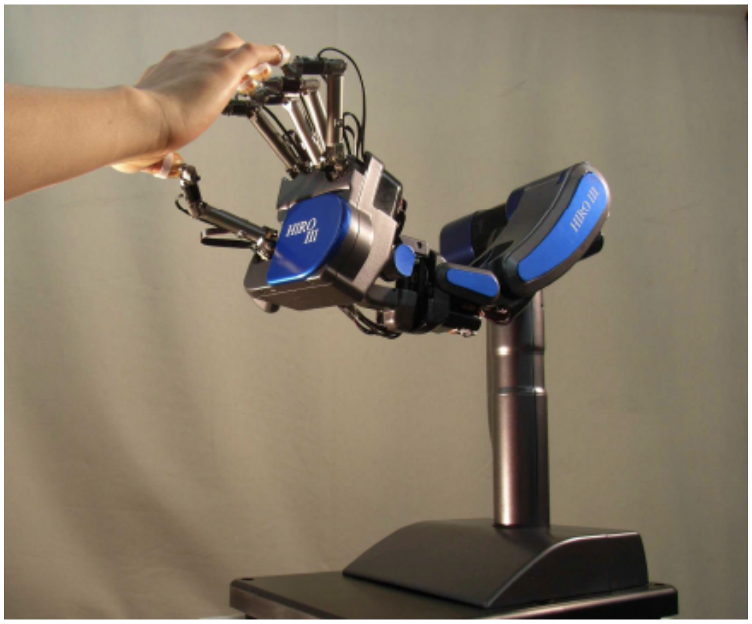


Figure 2. Hiro III.

(HIRO III 2008) presented opposed type five fingered haptic interface robot, so that it can simulate contact on the virtual world with the fingertips. The arm of this robot has six degrees of freedom and the total number of degree of freedom is fifteen (one for each finger joint). This system also provides with an API so custom development for this robot is easy and accessible. (Yoshida, et al. 2011) used this robot along with a 3D retro projector display to allow users touch a scene in a virtual co-located environment.

(Loscos, et al. 2004) created a system so that people can touch art pieces through a haptic device and additionally see them through a stereoscopic display. To achieve this they used a two contact point haptic device mounted on a exoskeleton with images projected on a CAVE like environment.

In summary, a significant amount of work has been done in order to provide a more immersive experience by completing different depth cues. Almost all of the work includes motion parallax and stereopsis in order to give a stronger sense of presence on the virtual reality. This approach provides better results when compared to traditional image displaying techniques, reducing the error from around twenty percent when using traditional techniques to sixteen percent or in some works mentioned above, this error reduces up to eleven percent. These results and systems are far from providing a fully immersive experience.

Our system was built so that we can display the binocular-stereo images in a very efficient way by implementing stereopsis through a stereoscopic display; motion parallax using head-tracking to feed it into a perspective correct rendering; and using a haptic device to provide tactile feedback to the user.

Our approach was different to other work because we use a depth map as input to encode visible geometry. This leads to some advantages, for example it can be easily encoded on the alpha channel on the binocular-stereo image or as a separate bitmap file; there are no shading calculations since the final color is already saved on the binocular-stereo image; and simplifies force calculations for the haptic device.

Additionally, one of our main goals in this system was to make it as accessible as possible to developers and users around the world. We emphasized the use of off-the-shelf, easily available hardware so individuals and institutions can use it for their own interests. The complete cost of the whole set of hardware and software used for this system keeps it below the one thousand dollars line. This also keeps its commercial availability for all kinds of purposes.

Chapter 3: Overview of the proposed system

The use of stereoscopic images on consumer devices had increased significantly during the last few years. Today, there are all kinds of electronic devices that includes stereoscopic displays such as televisions, videogames, or even cell phones. However, users are often limited to enjoy 3D imaging merely by staring to the screen having no interaction with the pictures shown.

The main motivation behind the system described in this thesis was to create a more immersive experience when watching 3D pictures. This system allows the user to see a 3D picture using a stereoscopic display, to synthesize novel point of views by merely changing the position of his head and to touch the picture through a haptic device.

The input data for this system is a simple depth map. This depth map can come either from a rendering system or from a horizontal disparity map calculated from a binocular-stereo image. To accomplish this, some state of the art algorithms were implemented along with a set of custom algorithms. The user interaction data was captured by off-the-shelf hardware in order to ensure that our experiments can be easily replicated and extended.

Chapter 4: Parts of the system

As seen on Figure 3, our system processes are divided into 4 different categories according to the task which they do:

- **Raw Input.** These modules get their input from two main sources, directly interfacing with hardware or from some files in order to accomplish the required task. In our system, for example the haptic device and the depth camera modules interface directly with the hardware. On the other hand, the rendering system module and the disparity map modules use external data, like a 3D scene description or binocular-stereo images, as input.

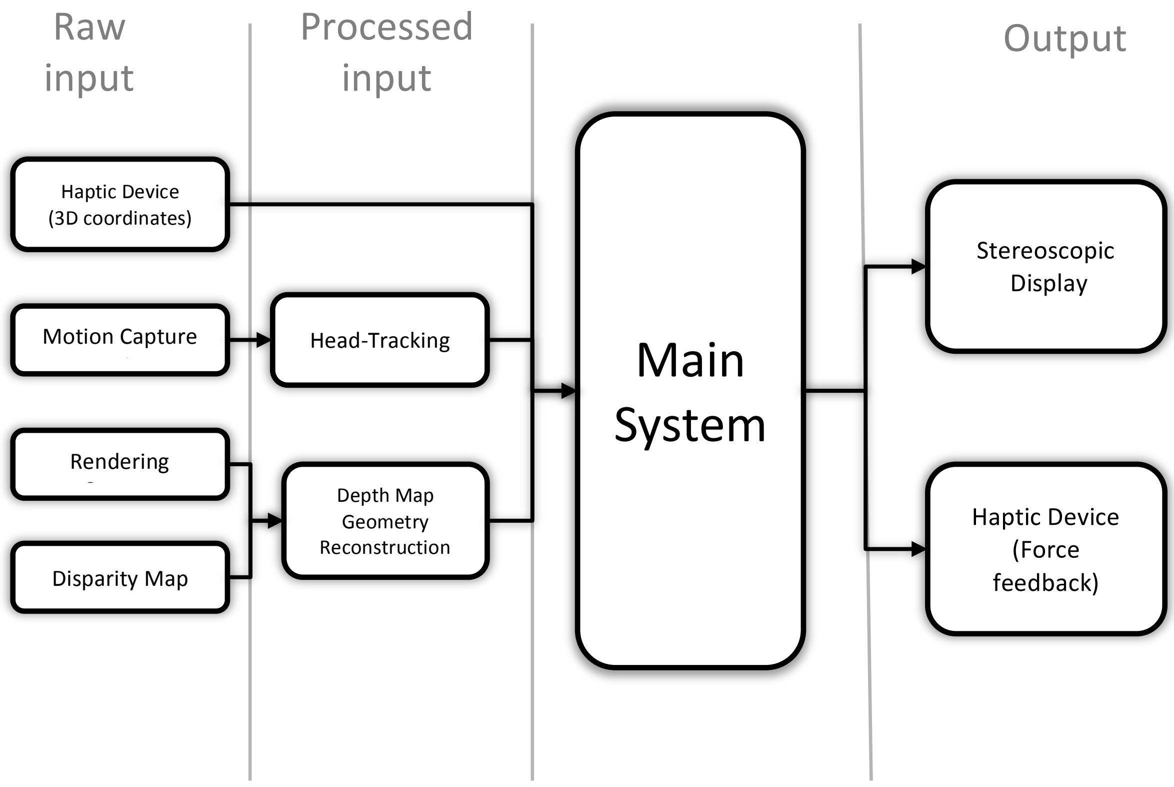


Figure 3. Diagram showing the main modules of the system and its data flow.

- **Processed Input.** These modules get their input from the raw input processes and provide digested data to the main system. The head-tracking module takes the input from the depth camera, runs some analysis over the data and provides the main system with a XYZ coordinate representing the position of the head relative to the camera. The depth map geometry reconstruction module on the other hand takes a depth map as an input and outputs a 3D mesh representing the scene ready to be sent to the main system..
- **Main System.** The main system is in charge of controlling and coordinating all data coming from the input devices and processing it so output modules can consume display the processed data.
- **Output.** It takes the processed data from the main system and sends it to the hardware that is going to present it to the final user.

During the rest of this chapter the author will describe the main parts and concepts of the system.

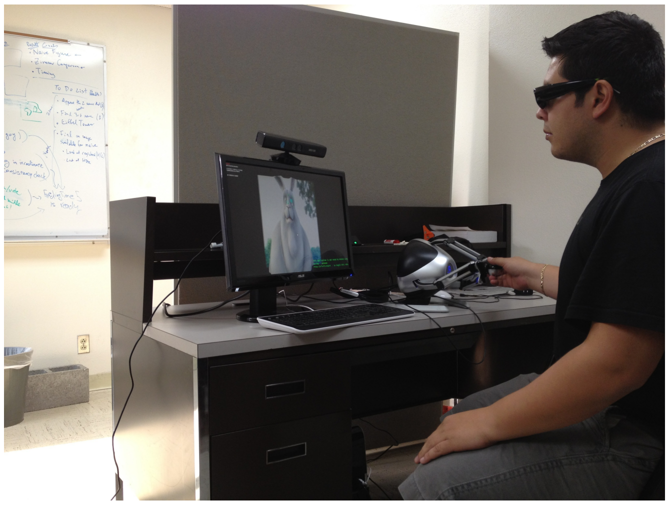


Figure 4. A user using the system. You can see the many hardware parts that conform the system: the Microsoft Kinect on top of the 3D Screen, the user wearing 3D glasses and the Novint Falcon.

Chapter 5: Generating the depth map

This system makes use of a depth map in order to give volume to a flat monocular or binocular-stereo image. In computer graphics, people often refer to a depth map as a bidimensional discrete image (bitmap) which describes the relative distance from the viewpoint where the image was rendered to the visible elements on the scene for each pixel (Computer Arts 3D world glossary 2011). It is often helpful to think about the distances stored in the depth map as an discrete representation of a normalized value between zero and one, where typically zero represents the minimum value on the depth map (and that is farthest away from the camera), and one is the maximum value which represents the closest point to the camera. You can see an example in Figure 5.

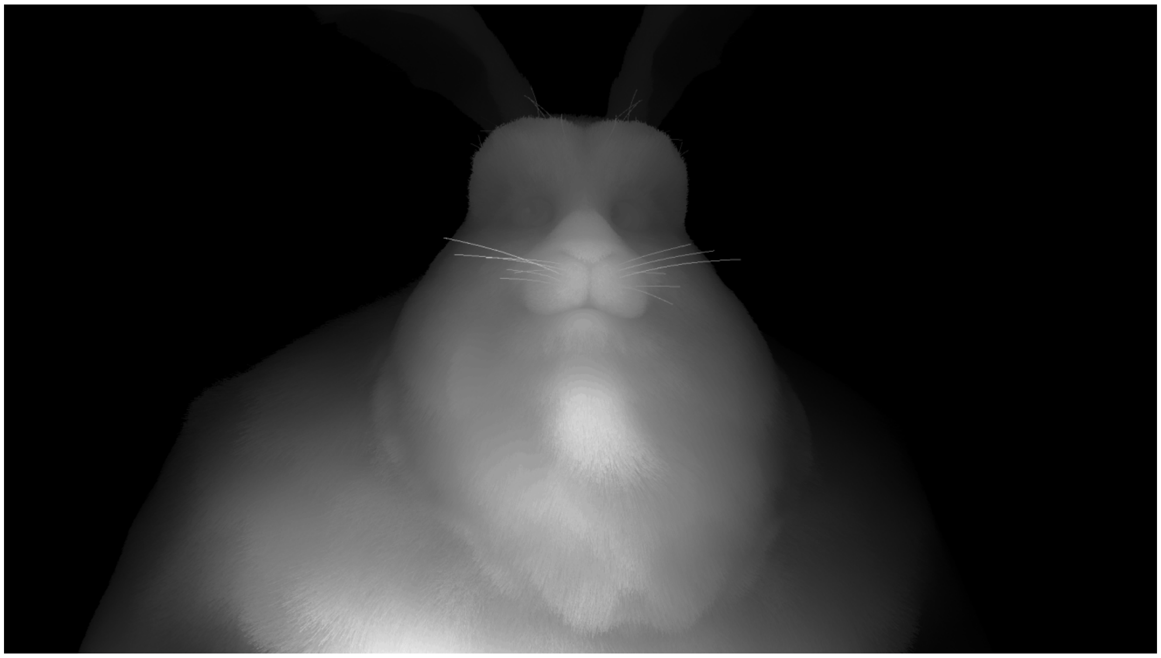


Figure 5. Depth map generated from the Big Buck Bunny movie (Goedegebure 2008).

Depth maps can be obtained in many different ways. The two approaches that we explored in this system are obtaining it directly from a rendering system and obtaining it from the real world by calculating it from a disparity map from a binocular image. We tested a number of algorithms in both categories in order to validate our approach.

The act of calculating a depth map is embedded in the heart of most of the rendering systems. In ray tracing rendering systems for example, we can calculate a depth map by saving the distance to the closest first intersection for each pixel in the image plane if we are only using one sample per pixel. If the scene is being sampled using some kind of Monte Carlo integration we should do a summation among all the relevant samples at each pixel at the image plane.

Open GL and Renderman systems calculate a depth map automatically in order to discern between hidden surfaces and surfaces that must be visible in the final rendered image. The depth buffer, also know as Z buffer, is written after the fragment program executes and can be found in the frame buffer object. So in the case of rendering systems it is easy to extract the necessary depth map.

In this study, we also explored getting a depth map from images captured in the real world. There are many algorithms known to recover the depth from the real world. They range from recovering depth from defocus (Chaudhuri and Rajagopalan 1999), using coded aperture (Levin, et al. 2007), using the camera motion (Williams 1980) or even using structured light (Scharstein and Szeliski

2003). All of them require specialized hardware and only work only in certain conditions.

(Chaudhuri and Rajagopalan 1999) analyze methods for recovering depth information by using the degree of defocus in an image. Since the degree of defocus is dependent on the lens settings, if we know those parameters the amount of blur can be estimated. Hence, it is possible to recover the depth. Its results are comparable in quality to the results yielded using disparity map algorithms, but it is often used when disparity analysis produces poor errors due correspondence errors and does a poor job as the aperture approximates to a zero.

The main draw back from this algorithm is the need of very high quality camera lens and that the lens configuration must be known beforehand. It is also very dependent on the image spatial resolution. This approach yielded poor results on images where real and rendered footage are combined when an image is composited with layers from different cameras and lenses or images with high amount of post processing. These types of images are often found on commercially available blockbusters movies, which were one of the inspirations since the inception of this thesis work. We decided not to use this method because of its limitation of producing good results on composited images.

(Levin, et al. 2007) proposed a very similar approach. This time they inserted a patterned occluder within the aperture of the lens so that the kernel where the out of focus elements get blurred (often called circle of confusion) is known and fixed. This helped to build a better and more accurate depth recovering filter for better quality depth maps. Even though (Levin, et al. 2007) proved to have better results than traditional depth from defocus algorithms and other disparity maps algorithms, the fact that the camera lens has to be physically altered limits its availability on common hardware and existing images.

Depth from motion (Williams 1980) uses a similar principle to disparity maps but instead of using a binocular-stereo image to find correspondences this algorithm finds them on monocular images across the time dimension when the camera moves. The results for this kind of algorithms are very dependent on the ability to solve for the movement of the camera and the scene contents. It is also hard to formulate solutions when movement of the camera and movement of the elements of the scene happen at the same time. Since one of the main goals in this thesis is to make every piece of the pipeline as general as possible, these algorithms are not useful in the general case because they fail when dealing with static images or when the contents of the scene are moving aligned with the camera. It also requires bigger efforts of maintenance to fine-tune the algorithm for different scenes.

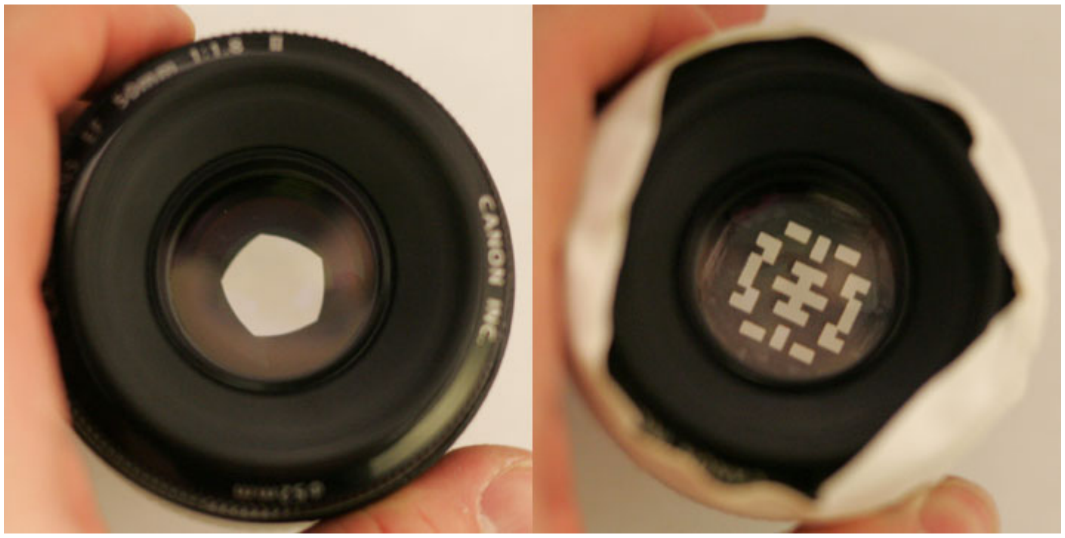


Figure 6. Normal aperture and coded aperture (Levin, et al. 2007)

In 2003 (Scharstein and Szeliski 2003) introduced to the community the idea of recovering depth using structured light. Their algorithm consists of shining different patterns of structured light from one ore many projectors to the scene. This effectively labels each pixel with a unique code, so that finding its correspondence in a binocular-stereo image is mostly trivial. The authors of this work acknowledge in their paper, that due to the intrinsic nature of this algorithm it is mostly useless for real life cases. Moreover, they bounded the contributions of their work as merely a tool to produce ground truth depth maps which would be used to compare again the results of different disparity map algorithms. Because this algorithm requires a structured light source (like a projector or a laser) is not commonly used on commercial works, mainly due its intromission on the pipeline by requiring to set up the system. Even though commercial systems are catching up quickly, such as the Microsoft Kinect, which uses invisible structured light (Microsoft Corporation 2011), they are not available in many devices yet and there a few professional productions that use them.

Different stereo disparity maps algorithms were also implemented for this thesis to evaluate its viability. Essentially, a disparity map tells the position differences among features in a pair of images (Okutomi and Kanade 1993). In the computer vision community, they are often encoded in bitmaps where each pixel value indicates a shift either on the horizontal or vertical axis.

The human vision system exploits this phenomenon to provide us with depth perception. Each of our eyes captures a bidimensional projection of the world from a slightly different viewpoint. The brain then extracts the depth information by finding feature shifts on the projections. This phenomenon is called stereopsis (Wheatstone 1838).

A typical stereo camera configuration is given in **figure 7**.

There are two different cameras that take a picture from slightly different view angles. Since a disparity map is essentially a shift of position of feature in a pair of images along an axis, both images have to be coplanar. Disparity in this case is horizontal. D is the separation among each cameras, and it is parallel to the X axis. Then, if *f* is the camera focal length and *d* is the disparity of the object point projected on each picture, the depth can be calculated as following (Chaudhuri and Rajagopalan 1999):

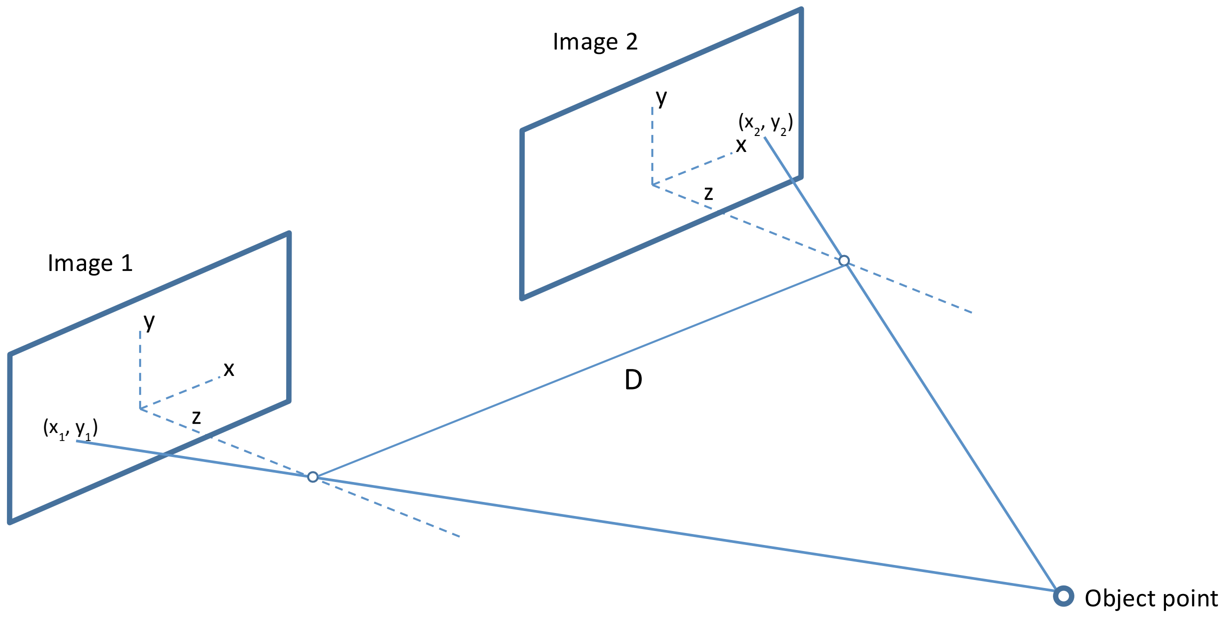


Figure 7. Stereo camera rig (Chaudhuri and Rajagopalan 1999).

$$z = \frac{f(D-d)}{d}$$

Binocular disparity map calculation is one of the most active areas in computer vision research; therefore many different algorithms have been developed. Local (window based) algorithms, such as normalized cross correlation (Hannah 1974) and rank transform (Zabih and Woodfill 1994), often only use intensity values of neighboring pixels subject to a finite window, making implicit smoothness assumptions. On the other hand, global algorithms make explicit smoothness assumptions in order to solve the optimization problem that minimizes a global function that combines data and smoothness terms (Scharstein and Szeliski 2002). Examples of those kind of algorithms are the ones based on graph cuts (Boykov, Veksler and Zabih 2001) or simulated annealing (Barnard 1989).

(Marr and Poggio 1976) described a cooperative algorithm that uses associated local constraints to calculate disparities among features on images. (Hong and Chen, Segment-based stereo matching using graph cuts 2004) presented a segment based on stereo matching algorithm that poses the problem as an energy minimization problem, for which they used graph cuts to solve it. For each graph cut a disparity plane is given.

For our system we decided to use a cross correlation based algorithm (Cochran and Medioni 1992). They described an algorithm in which they used an area based and feature based primitives to match pixels among each point of view in a binocular-stereo image. The area-based processing uses cross-correlation along with an ordering constraint to provide with an initial smooth and dense disparity map. Due to the iterative nature of this algorithm, details are introduced in the disparity map when using the featured based primitives on next iterations on a Gaussian Pyramid built from the original binocular image. This algorithm proved to provide a smooth disparity map when preserving details in our experiments.

Disparity maps also have some drawbacks. The process of creating the disparity map is very expensive because of the intrinsic nature of the algorithm of looking for corresponding features on both images. There is also an inevitable introduction of noise while calculating the map, so there is a need of the introduction of denoising algorithms to have a smooth map.

For the sake of simplicity and with the intention that our system can be used for mass consumption, we chose to calculate the depth map by using binocular-stereo images and calculating the horizontal disparity map (Cochran and Medioni 1992).

A disparity map can be directly translated to a depth map by normalizing its values so that it can be used on a rendering system. This posed a great benefit: it allowed us to use commercially available hardware to capture binocular-stereo images to be used in our system or simply using data available on binocular-stereo 3D movies.

Since our system only receives a depth map, it is totally decoupled from how it was generated. The system is fed with only a image containing the depth information. Common file formats have a color depth that varies from 8 up to 32 bits in most cases. In our experiments we found out that often the depth maps had more resolution than the haptics devices can handle. Therefore we used some detail mapping algorithms in order to fit the geometry encoded in the depth map in the limited resolution of the haptic device.

A simple naive normalization of the depth values produces poor results since scene detail is often lost because its values get mapped into a single bin on the normalized histogram. To avoid this artifact we decided to implement and evaluate a number of detail mapping operators so the high dynamic range of values on the depth map can be mapped to a more limited resolution without having to compromise the perceived quality of the geometry reconstruction.

Effectively a detail mapping algorithm reduces the contrast from the scene value to an adequate range for the containing medium. There is a good variety of detail mapping algorithms. Some of them are used only for producing aesthetically images, while some others try to reproduce the image as close as possible to the original. In Figure 4, there is a comparison between the original depth map from a rendering system and the detail mapped depth map, so that details can be rendered on the haptic device.

For this project we had special constraints since we were dealing with a depth map rather than a common image. In this case the pixel values in the bit map represent the distance from the camera to the object rather than color. Maintaining a similar derivative across the pixels is important since this depth map is being used as an input to reconstruct the geometry using a mesh. So, in this project we were looking for a detail mapping algorithm that allow us to compress the values while maintaining relative contrast among pixels rather than an algorithm that produced aesthetically results, therefore we focused on local detail mapping algorithms. These detail mapping algorithms are in essence very similar to what we call tone mapping algorithms.

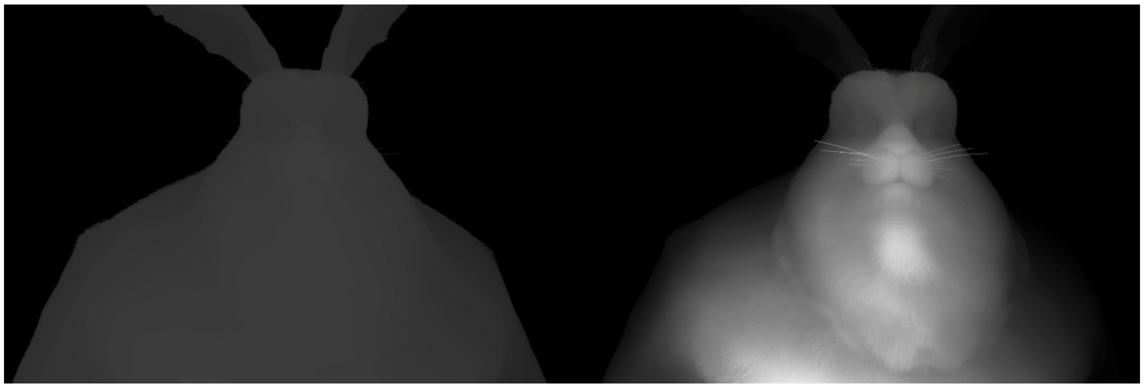


Figure 8. The bunny on the right is the depth map straight from the rendering system, there is little detail that can be render in a device with limited resolution. The image on the right is the depth map after being detail mapped; the details can be appreciated with the naked eye.

The most basic detail-mapping algorithm consists on merely normalizing the values on the original image and scales them back to an appropriate depth color resolution for the containing medium or applying a function such as a cubic root or a logarithm to the luminance values. This algorithm is very naïve and often ignores image contents. If we merely normalize and scale back the values to a quantized grid, there is a big risk of losing scene details because several different values on the original image can be map into a single bin in the histogram of the tone map image.

(Chiu, et al. 93) proposed a local spatially non uniform algorithm to do tone mapping that uses normalization as base of its algorithm. Then they applied a gamma correction and error minimization algorithms based on gamut mapping (Glassner, et al. 1993) and brightness mapping (Tumblin and Rushmeier 1991). This algorithm was originally intended to display high dynamic range pictures on the old monochromatic display that had a resolution of only 4 bits. However it can be easily extended from 8 to 32 bits or more without any problem. This algorithm does a good job while preserving detail in the low dynamic range image, but produces halos on regions near to high frequencies.

(Durand and Dorsey 2002) presented a technique that reduces the contrast of the image while preserving details. To achieve it, they do it by doing a two scale decomposition of the image by separating the base from the detail of the image. Only the base got its contrast reduced. They do it by applying a bilateral filter, which is an edge preserving filter. This is a non linear filter where each pixel, in the spatial domain, is weighted by a Gaussian and if big differences appear in its derivative, the value of this Gaussian is decreased. This tone mapping algorithm effectively lowers the contrast among the base and keeps detail in the image without any strange artifacts like banding or halos.

Since in our system uses a depth map which is encoded as a grayscale image, the tone mapping algorithm to use only has to deal with luminance. Due to its simplicity, we decided to use the (Durand and Dorsey 2002) along a tone mapping tool that allows us to specify a custom function in order to weight the different bins in the histogram, while using the bilateral filtering to preserve detail. We provide more detail about our implementation in chapter 9.

Chapter 6: Stereoscopic display

Typically displays like the ones we typically find in our offices or cell phones can only show bidimensional images without any depth, because they project exactly the same image to each of the eyes. Much engineering effort has been put in order to create displays that create three dimensional images in where users can perceive depth.

(Wheatstone 1838) created what probably was the first 3D display of the history. He created a head mounted device, similar to today's glasses, which presents a slightly different image in front of each eye, effectively multiplexing spatially the binocular-stereo image, in order to prove the phenomenon of stereopsis. At that time, 1838, he used hand drawn pictures; nevertheless people soon started using binocular-stereo pictures with the advent of photography.

There are many techniques to build stereoscopic displays that vary from using a stereoscope (Wheatstone 1838) such as free viewing (Erker 2011), cross viewing (Erker 2011), head mounted displays, polarizing or filtering glasses (Jorke and Fritz 2006), color anaglyphs (Gernshein and Gernshein 1969), lenticular prints, and parallax barrier (Isono and Yasuda 1994).

Free viewing and cross viewing (Erker 2011) multiplex a binocular-stereo image spatially as well. These techniques present a binocular-stereo image, on which each viewpoint is presented side by side in the horizontal axis, and then ask the users to correctly focus and orient their eyes to be able to perceive depth on the image. Free viewing, also known as Parallel Stereo, requires users to focus on a nearby plane while orienting their sight on the far infinite, so that when watching the binocular-stereo image just in from of them, the left image projects to the left eye and the right image projects to the right eye. Cross viewing, on the other hand, does exactly the opposite. Users must focus on a far plane while orienting their sight on a nearby object. In this way the left image gets projected to the right eye and the right image gets projected to the left image.

Free viewing and cross viewing do not require the use of glasses and can be implemented in any existing display hardware as long as the resolution allows it, since these techniques are merely multiplexing the image on the spatial domain. On the other hand, these techniques have a steep learning curve and often produce fatigue and discomfort to most of the users.

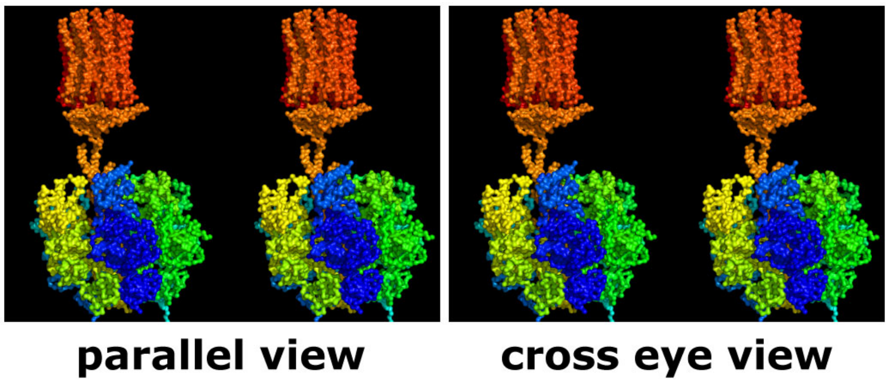


Figure 9. Protein visualization rigged to perceive depth according to the indicated technique (Del45 2009).

Anaglyphs (Wihelm 1853) have been a very popular technique since the early sixties to deliver a stereoscopic 3D effect. This technique consists on having a color-coded picture in which each of the viewpoints is rendered using chromatically opposite color schemes. These images are later revealed by using glasses where the lenses are also chromatically opposite, so that they filter out their corresponding layer on the display device letting go through a different image for each eye.

One of the most popular color schemes for anaglyphs is the use of red and cyan, often using red for the left eye and blue for the right eye. This technique is one of the most widespread for stereoscopic vision and has been widely used in the last fifty years, mainly because of its low cost of implementation and the fact that high-end glasses are not required for most of the applications. There are implementations where the image is just printed in plain paper and the glasses are built using cardboard and inexpensive tinted plastic for the lenses. The original technique by (Wihelm 1853), which uses red and cyan color schemes, poses a limited color rendering ability although it is the most commonly used. Few other chromatic schemes have been proposed in order to provide better color rendering abilities.

ColorCode 3D (Sorensen, Hansen and Sorensen 2001) uses and amber (red + green + neutral gray) and a pure dark blue lens along with optional correction lens. It offers a much better color rendering than traditional anaglyphs (red and cyan) but produces a dark image, therefore a dark room is needed along with a very bright image. Recently, on 2007 TriOviz (TriOviz 2010) introduced the INFICOLOR anaglyphs. TriOviz used a patented color schemes that was called complex magenta and complex green. This anaglyph system can be used with traditional bidimensional screens and television sets that allows the perception of natural and vibrant colors. Also when observed without glasses a slight and almost unnoticeable doubling can be noticed on the background.

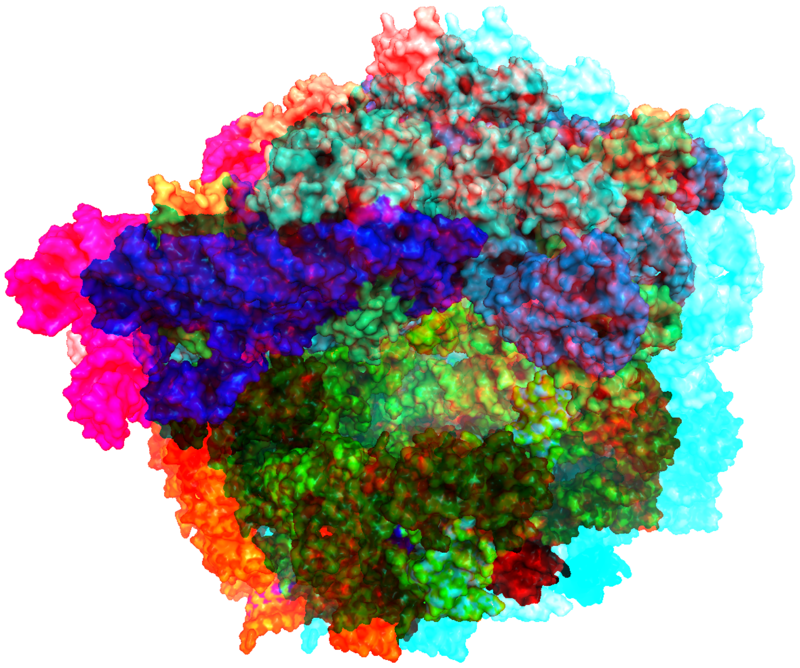


Figure 10. Anaglyph representing the visualization of a Ribosome. This Anaglyph is intended to be used with red and cyan glasses (Vertrees 2011).

Parallax barrier (Berthier 1896) consist of a partial occluder put in front of an interlaced image source, such as a liquid display. This occluder is composed of a series of precision horizontal slits, allowing when the head is in the correct position to see a different set of pixels when the head is in the correct position. This solution is very inexpensive to build, because it merely adds a precision occluder that sits in front of a bidimensional display. The greatest drawback of this technique is that it requires a fixed position of the viewpoints in its most basic case (two images). If broader view angles are needed a bigger amount of interlaced images are required. For example a Toshiba 21 inch 3D display that uses the parallax barrier technology needs 9 pair of images to cover a viewing angle range of 30 degrees (Toshiba 2010).

Lenticular displays (Lippmann 1908) use a thin layer of contiguous lenses over a bidimensional display to render project a different image according to the viewpoint. There are two different schools of thought for 3D displays using lenticular displays. The first one uses the lenses on the lenticular layer to show a different set of pixels for each eye; it is similar to the parallax barrier method.

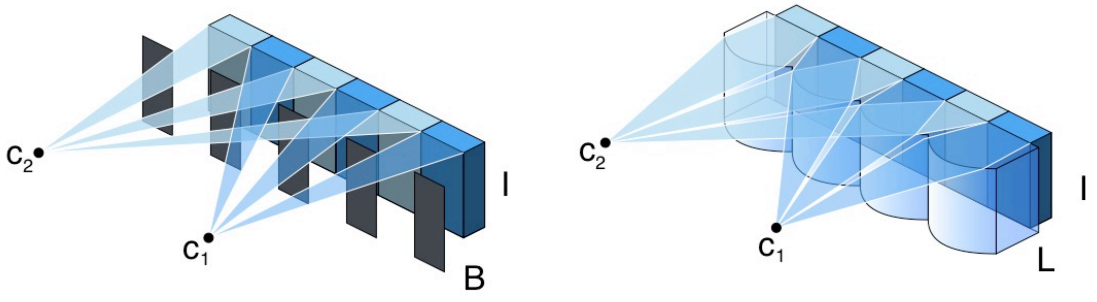


Figure 11. Comparison between the parallax barrier method and the lenticular screen method. Where c_1 and c_2 are the cameras viewpoint, B is the parallax barrier, L is the lenticular screen and I is the interlaced binocular-stereo image (Cmglee n.d.).

The second, is lenticular displays. It is an array on lenses that are put on top of an interlaced image allowing the user to perceive the stereopsis. This process consists of two steps. First, the images to be projected to be used must be interlaced in a bidimensional image so that the lenses layer can help to deinterlace the image according to the viewpoint. Even though the process can display 3D images without the need of any glasses it is usually a very expensive solution. The cost of these displays actually increases according to the number of viewpoints they can render; nowadays commercially available displays nowadays can render up to 64 viewpoints. Content bandwidth is also a big problem with stereoscopic displays, since an underlying resolution is needed for the interlaced image of the desired resolution times the number of viewpoints.

There will always be a tradeoff among the desired resolution, the number of viewpoints and the resolution provided by the underlying display for the interlaced image. For example, if you want to render sixty four viewpoints at a HD resolution (1980 x 1200 pixels) you will need an underlying resolution of at least 126,720 x 76,800 pixels. In 2011, having a device able to process and display that amount in pixels is rare and very expensive.

We decided to use a Liquid Crystal Shutter Glasses solution provided by Nvidia (Nvidia 2011). This solution works by using an alternate frame sequencing imaging system on a 120 Hz LCD monitor synchronized with a pair of Liquid Crystal Shutter Glasses that have the property of darken when a current is applied working at the same frequency. This imagining system renders an alternate sequence of binocular-stereo full size images from each of the viewpoints for the binocular-stereo picture, so that the shutter glasses only allows to see one eye a viewpoint, and then allows the other eye to see the image rendered from a different viewpoint. This effectively produces the phenomenon of stereopsis when refresh rates on the display are greater than 60 Hz. Although a minimum refresh rate of 120 Hz is recommended to fully eliminate flickering.

This method eliminates the ghosting problem often shown in other methods, such as anaglyphs or with polarized glasses. However, since each lens is effectively darken half of the time, the luminance perceived from the display is dimmer.

For the Nvidia 3D vision solution, the implementation was straightforward. The video card renders automatically the binocular-stereo image when using the Direct X framework or it can it can be also custom coded by developers if they use the proprietary NVAPI by using the quad buffered mode. In this way developers can either choose how the binocular-stereo image gets rendered or control different parameters, such as the screen depth (convergence) or stereo separation. The shutter glasses use a wireless IR protocol to synchronize them with the refresh rate of the monitor.

This process will help with the adoption of the system. By choosing a standard, mass consumer oriented hardware we ensure an easy-to-replicate architecture that will allow people around the world to use this system as a base to implement new functionalities.

Head-tracking refers to the act of registering the 3D position of the head of a user at any given time. In our system, we use the head-tracking functionality to correctly render image to the user perspective. In order to do this we simply capture the position of the head and then use this information as input parameters to calculate the position of the camera on Direct X. Since the units of the head-tracking system differ from the ones we used on the Direct X application, we must do some calibration so that the coordinates transformation mapping are coherent.

Much effort in the research community has been invested to develop a wide variety of head-tracking algorithms under different approaches. (Black and Yacoob 1995) explored the use of optical flow to build a model in which they follow rectangular planar patches defined by the different parts of the face, to later calculate their affine transformation. (Azarbayejani, et al. 1993) and (Jebara and Pentland 1997) used feature point tracking to locate distinctive parts of the face, like the corners of eyes or mouth. Skin color was also used by (Yang, et al. 1998) to create an adaptive stochastic model.

Many other researchers have also used textured geometry as a mean to track head position. (La Cascia, Isidoro and Sclaroff 1998) used a textured cylinder as a head model while (Schödl, Haro and A. Essa 1998) used a polygonal head model that is matched with the incoming video stream. Stereo reconstruction, (Yang and Zhang 2002) and (Newman, et al. 2000), is one of the most accurate systems to do head-tracking, mostly because of its native tolerance to occlusion and redundancy of information. It recovers the geometry from a multipoint camera system to find face features on the 3D space.

Following the philosophy of using off-the-shelf and commercially available hardware, we decided to use the Microsoft Kinect (Microsoft Corporation 2011) motion capture device. This hardware was originally conceived to be used with the Microsoft Xbox 360 (Microsoft Corporation 2011) to provide a controller-free gaming experience. It uses a special combination of hardware and software in order to provide multi user motion capture capabilities. Moreover Microsoft Research has developed an easy to use and very well documented API which it has been made available to developers (Kinect for Windows 2011).

The Microsoft Kinect works by using an infrared random laser grid projected on to the scene combined with a CMOS infrared sensor and a color CMOS sensor. Then, the hardware then sends an 8-bit 640x480 pixels depth video stream along with a correlated 24 bits 1600x1200 pixels color video stream at 30 frames per second through a proprietary USB connection to the computer to be analyzed (Shotton, et al. 2011). Additionally, it also holds two analog microphones along with an array of 4 digital microphones.

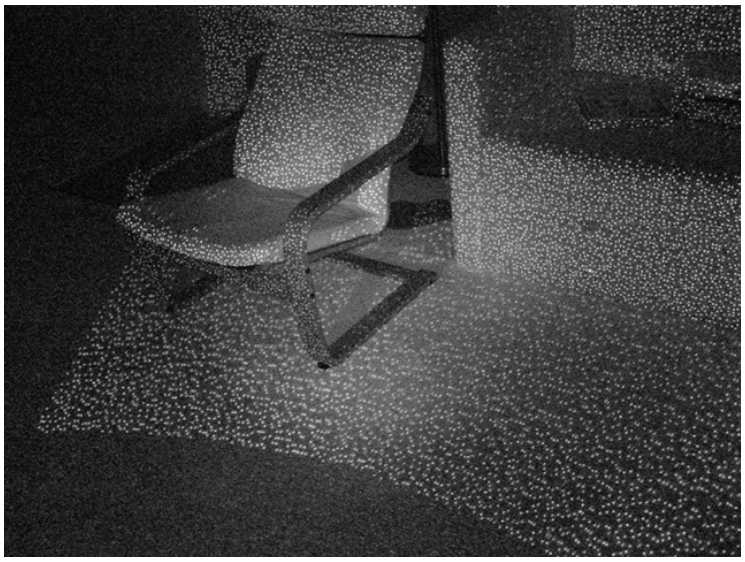


Figure 12. Infrared picture showing the structured light projection.

The idea behind generating a depth map from structured light is simple. If you have a light source with a certain offset on the projector side, the spot of light is shifted according to the distance it is reflected back from when the light beam gets projected into the scene. So if you can track every spot on the projected infrared random laser grid you can have a good reconstruction of the scene geometry. The Kinect does this geometry reconstruction in a 11 bit depth map and it is done directly in the Kinect hardware by the Prime Sense PS1080-A2-SoC image sensor processor (Freedman, et al. 2010). This allows the system to work in low light conditions, resolves silhouette ambiguities and provides a calibrated scale estimate along with a color and texture invariant result.

Once the video stream is received at the host computer, the drivers analyze the incoming depth data to provide accurate motion capture data. (Shotton, et al. 2011) solved this problem by treating it as an object recognition problem. They first segment the depth map, without using any temporal clues, into a dense probabilistic body part labeling, with the parts defined to be spatially localized near skeletal joints of interest. Then they reproject the inferred parts into world coordinates to localize spatial modes of each part distribution to generate a confidence-weighted proposal for the 3D locations of each skeletal joint.

For the training data, they relied on a large database of synthetic images of human in many shapes and sizes in varied poses, which were gotten from motion capture of human actions. This database is latter used to train a randomized decision forest classifier with thousands of training images. This approach have many advantages over state of the art approaches, since it's discriminative approach naturally handles self occlusions and poses cropped by the image frame. It also provides results that are very accurate and stable because they do not have any temporal dependency.

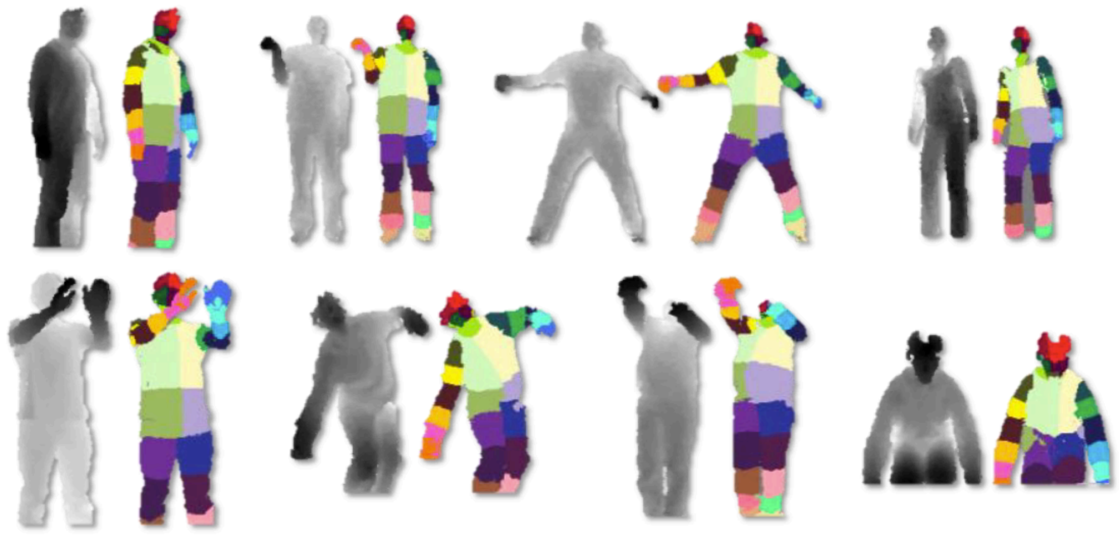


Figure 13. Body part labeling results from the Kinect. Excerpt from (Shotton, et al. 2011).

All of this computation is done automatically by the Prime Sense drivers and Microsoft Kinect libraries. The developer main task while using this data relies merely on initializing the hardware, querying for a specific joint (the head in this case) and realizing any object created at run time when the application ends being used.

Chapter 7: Haptics

Haptic technology, or haptics, refers to a set of tools and techniques that provides the final users the sense of touch by providing tactile feedback by applying forces, such as vibrations or motions when interacting with virtual environments. It is often described as “doing for the sense of touch what computer graphics does for vision” (Robles-De-La-Torre 2009). The act of providing information on how to deliver tactile sensations to the user is often called haptic rendering.

Mainly, there are mainly two types of haptic devices: One-way haptic devices that only provide tactile feedback, like the cell phone vibrator motors.; and two-way haptic devices, that in addition to provide tactile feedback also works as an input devices, often in a form a pointer, such as the Phantom devices (Sensable 2011).

Essentially a haptic device consists of a programmable motor which produces forces that can be sensed with the touch. There are many different types of underlying technologies that power the haptic devices, including electric motors like (Roberts, Slattery and Kardos 2000) and (Bach-Y-Rita, et al. 1969), pneumatic pressure (Yobas, et al. 2003), electric actuators [(Sensable n.d.), (Kawai and Yoshikawa 2000)] and electro vibration based (Bau, et al. 2010). The haptic devices also vary on the degrees of freedom that they can have. On one hand for example, (Yoshikawa, et al. 1995) created a one-degree-of-freedom impedance display that can simulate friction while interacting with virtual objects.

On the other hand, there are very complex haptic devices such as the multi-degree of freedom hand exoskeleton (Avizzano, et al. 2000). In which every finger has three degrees of freedom in correspondence to the human finger flexion axes.

Haptic rendering is also a very important part of the haptic system. Most haptic rendering systems use interpolated normals to calculate the force vectors to be used by the haptic device. (Morgenbesser and Srinivasan 1996) proposed a model analog to the phong shading algorithm, which refers to a controlled variation on the surface normals in order to give the illusion of touching a non-flat continuous surface on a nominal flat surface. This can effectively reduce computation times because there is no need of a high resolution model in which collision detection must be implemented. Instead we can use a low resolution model and user's perception would be equivalent. (Hayward and Yi 2003) proposed a similar method in which surfaces normals are needed and instead it calculates forces change of height among vertices. (Avila and Sobierajski 1996) proposed quantizing surfaces and models into touchable voxels in order to easy collision detection.

In this project, we needed a bidirectional haptic device that could serve as a 3D input on our system as well to provide feedback. There are many different haptic devices commercially available that would fit our technical needs. We found several devices that can accommodate our needs such as the Sensable Phantom (Sensable 2011). This device is mostly used for high-end applications on research laboratories and high end training programs, like surgeon training.

The cost of this product is over USD $30,000, which makes it prohibitive to the end user mass market.

For this project we decided to use the Novint Falcon (Novint Technologies 2011). The Novint Falcon is a bidirectional haptic device that uses 3 arms to position a detachable handle on a 4 inches by 4 inches by 4 inches 3D touch workspace, a position resolution of 400 dot per inch, a force capability of up to two pounds and a refresh rate of 1 kHz and connected to a host computer through a USB connector (Novint Technologies 2011). Novint Technologies provides with a Software Development Kit (SDK) to easily implement any kind of haptic interaction with the Novint Falcon. Although the technical specifications of this device seem to be lower than the Sensable Phantom, its price is more affordable and hence easily accessible for the final user, starting at only $250 USD. the Novint Falcon was the best device to use for this work, since our goal was to create a system under one thousand dollars, with off-the-shelf, easily accessible hardware.

Now that we have described the stages of the proposed system, in chapter 8 we are going to describe in detail the main algorithms that we use for our experiments.

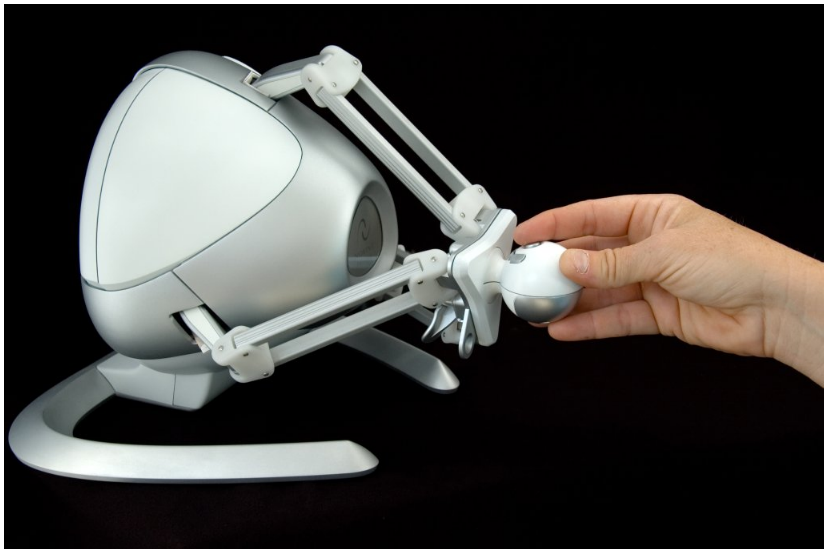


Figure 14. The Novint Falcon (Novint Technologies 2011)

Chapter 8: Disparity Map

One of the most important parts of our system is to obtain the geometry from the scene to be rendered by feeding the system with a depth map. For example, rendering systems can produce a depth map that describes distance from the image plane to the nearest objects in the scenes. This depth map is easily obtained with rendered images. In this section we will describe how to obtain it from a binocular image.

To obtain depth information from the real world the research community has proposed many different approaches and systems. You can find a survey of these approaches in the introduction chapter of this thesis. One of the most reliable ways to capture depth information from the real world is by calculating the horizontal disparity map from a binocular-stereo image. This essentially calculates the distance of a feature on the left image of a binocular-stereo image to the corresponding right image, on the horizontal axis. For this system, we decided to implement the algorithm of (Cochran and Medioni 1992) which will be discussed in greater detail further in this thesis.

Rectifying the epipolar line

Calculating a disparity map can be described as finding the Euclidean bidimensional distance in a plane among features presented on both images on a binocular-stereo image. Sometimes when capturing or rendering the binocular-stereo image, camera planes are not coplanar, and it increases the difficulty of finding feature correspondences among the left and right images.

One method to simplify feature correspondences search is by rectifying images so that they share the same image plane, which reduces the search space into only one dimension. Now feature correspondence search now has to be done in the horizontal axis only (corresponding rows of the rectified images), which are parallel to the baseline between cameras. Furthermore, since we know images are located either right or left from each other, finding a feature from the left image on the right image should be only done to the right on the horizontal axis of the right image and vice versa, reducing search time.

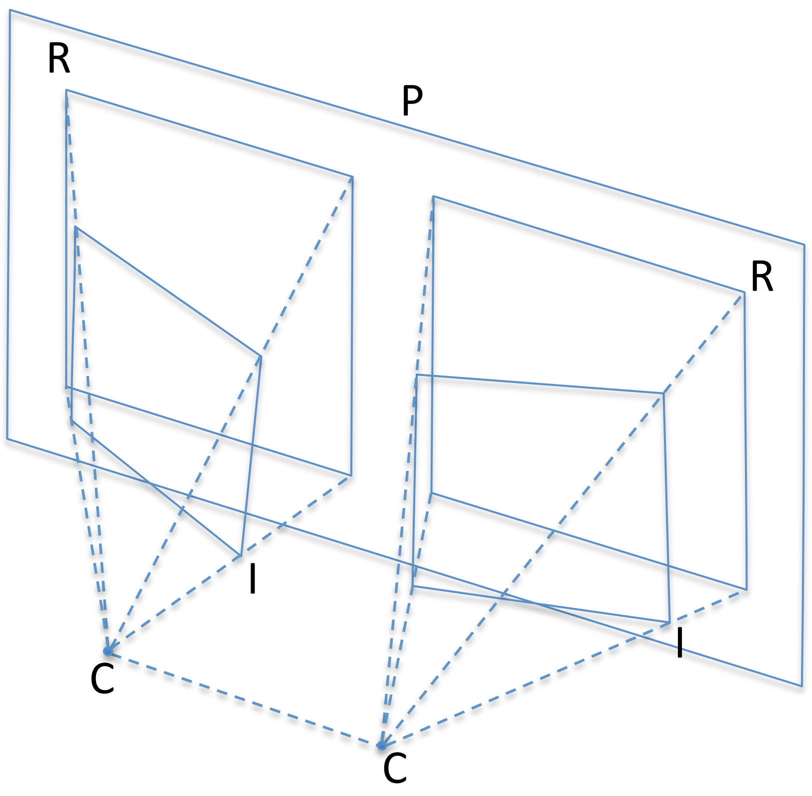


Figure 15. Planar rectification. (R_1, R_2) are the rectified images for the (I_1, I_2) pair. P is the image plane parallel to the baseline between cameras (C_1, C_2).

The standard rectification process is straightforward. It consists on finding a plane that is parallel to the baseline between the camera center points. Then we should find the transformations that reproject the images to this plane. This is illustrated on figure 5.

We used a similar method based on (Szeliski 2010). Since parallel lines remain parallel under affine transformations, we can recover, for the pair of the given binocular-stereo images, the affine properties by using the transformation matrix H_1, which maps the vanishing line into the line at infinity $\vec{l}^{\infty}$.

Given a pair of parallel lines for each image $\vec{l}^{(1)}$, $\vec{l}^{(2)}$ and $\vec{m}^{(1)}$ and $\vec{m}^{(2)}$, where $\vec{l}^{(1)} \parallel \vec{l}^{(2)}$ and $\vec{m}^{(1)} a\, \vec{m}^{(2)}$. This set of line on a perspective-distorted image will interest at the vanishing point $\vec{p}^{(1)}$ and $\vec{p}^{(2)}$. The line formed by connecting $\vec{p}^{(1)}$ and $\vec{p}^{(2)}$ is the vanishing line $\vec{l} = (l_1, l_2, l_3)^T$. So that if we have it in homogeneous coordinates:

$$\vec{p}^{(1)} = \vec{l}^{(1)} \times \vec{l}^{(2)},$$

$$\vec{p}^{(2)} = \vec{m}^{(1)} \times \vec{m}^{(2)},$$

$$\vec{l} = \vec{p}^{(1)} \times \vec{p}^{(2)}$$

Equation (1)

If we apply the transformation using the homography H_1, we get:

$$H_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{pmatrix}$$

Equation (2)

The vanishing line will be mapped in to the line at infinity

$$\vec{l}^{\infty} = (0,0,1)^T$$

Equation (3)

If we do the inverse transform of H_1, we can prove the correctness of equation (3).

$$H_1^{-T} = \begin{pmatrix} 1 & 0 & \frac{-l_1}{l_3} \\ 0 & 1 & \frac{-l_2}{l_3} \\ 0 & 0 & \frac{1}{l_3} \end{pmatrix}$$

And,

$$H_1^{-T}\vec{l} = (0,0,1)^T$$

Once we have calculated the affine rectified image I_a, we should remove the affine distortion, therefore we must find the affine transformation matrix:

$H_2 = \begin{pmatrix} A & \vec{t} \\ \vec{0} & 1 \end{pmatrix}$,

Such that,

$$H_a = H_2 X_s$$

where X_s is the image on the real world.

If we have a pair of orthogonal lines, $\vec{l} \perp \vec{m}$. $\vec{l'}, \overrightarrow{m'}$ will be the transformed lines under the affine transformation H_2. We know that:

$$\left(\frac{l_1}{l_3}, \frac{l_2}{l_3}\right)\left(\frac{m_1}{m_3}, \frac{m_2}{m_3}\right)^T = 0$$

then

$$l_1 m_1 + \ l_2 m_2 = \ \vec{l}^T C_\infty^* \vec{m} = 0$$

where

$$C_\infty^* = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

describes the dual degenerate conic. Since $C_\infty^{*'} = \ H_2 C_\infty^* H_2^T$, we have

$$\vec{l}^T C_\infty^* \vec{m} = \vec{l'}^T H_2 H_2^{-1} C_\infty^{*'} H_2^{-T} H_2^T \vec{m}' = \ \vec{l'}^T C'^*_\infty \vec{m}' = 0$$

Therefore,

$$\vec{l'}^T C'^*_\infty \vec{m}' = \ \vec{l'}^T H_2 C_\infty^* H_2^T \vec{m}$$

$$= \vec{l'}^T \begin{pmatrix} A & \vec{t} \\ \vec{0} & 1 \end{pmatrix} \begin{pmatrix} I & \vec{0} \\ \vec{0} & 1 \end{pmatrix} \begin{pmatrix} A^T & \vec{0} \\ \vec{t}^T & 1 \end{pmatrix} \overrightarrow{m'}$$

$$= \vec{l}^T \begin{pmatrix} AA^T & \vec{0} \\ \vec{0} & 0 \end{pmatrix} \overrightarrow{m'}$$

We have

$$(l'_1, l'_2) AA^T (m'_1, m'_2)^T = 0$$

In order to get A, let S = AA^T, where $S = \begin{pmatrix} s_{11} & s_{12} \\ s_{12} & 1 \end{pmatrix}$ because S is symmetric.

$$(l'_1 m'_1, l'_1 m'_2 + l'_2 m'_1) \binom{s_{11}}{s_{12}} = -l'_2 m'_2$$

Since S can be written as S = UDU^T, where $U^{-1} = U^T$, and we know that $A = U\sqrt{D}U^T$, we find that H_2 and the transformed image is $X = H_2^{-1} X_a$.

Disparity Map Construction

In its essence, a disparity map describes the difference of location among corresponding pixels of the same scene of images taken at different angles and it is often used to reconstruct scene depth information. The process itself is a difficult problem and it is ill posed. The process of finding accurately corresponding pixels on quantized images is very prone to find mismatches introducing noise to the final disparity map, specially on texture less regions and regions where disparity is discontinuous.

There are two different main ways to create disparity map. Window-based algorithms calculate what the disparity map is by taking into account only the intensity of pixels in a finite neighborhood or window. These kinds of algorithms produce very good results on images with high amount of textures, however they produce very noise results on regions with constant colors and tend to blur the disparity map where discontinuities are found. There are also global algorithms, which make explicit smoothness assumptions and treat the disparity map creation problem as an energy minimization problem.

For our experiments with the system we decided to implement the best algorithms from both kinds of approaches to evaluate their feasibility after dealing with real life images with problems such as some noise on the images due to normal errors found on consumer cameras, composited images and images produced by a rendering system.

The first algorithm we tried was a global algorithm (Hong and Chen 2004). For this global based algorithm we used graph cuts in order to find correspondences among the rectified images. In this approach, the reference image is divided into non-overlapping homogeneous color segments and the whole image is represented as a set of planes on the disparity space. Matches in this algorithm are done across different segments, rather than on the pixel level as in the window based algorithms. As previously mentioned, this algorithm poses the disparity map construction problem as an energy minimization problem, but this time on the segment space.

$$E(f) = E_{data}(f) + E_{smooth}(f)$$

More specifically the energies to minimize are the smoothness energy (E_{smooth}) that measures the disparity smoothness between neighboring segment pairs and data energy (E_{data}) that measures the disagreement of segments and their matching regions based on assumed disparity planes. The graph cuts technique is used to get a fast and approximate solution to the optimal solution.

One of the most important assumptions of this approach is that disparity discontinuities occur only on the boundaries of homogeneous color segments. Therefore any color segmentation algorithm can be used as input for this this approach. For the experiments of our system, mean-shift color segmentation is used.

It is important to state that each of the segments is treated as a disparity continuous surface or a plane. Planes are calculated as following: first the local disparity is obtained used a window based algorithm. Then the initial parameters are calculated for each color segment. Finally planes parameters are refined through fitting them to grouping neighboring segments.

Once we have color segmented the image and calculated local disparity planes, we should match corresponding planes using graph cuts. R is the color segments of the reference image and D is the estimated disparity plane set. The main goal is to find a match f that assign each segment $S \in R$ a corresponding plane $f(S) \in D$, where f is both smooth and consistent. Formulated as an energy minimization problem can be written as follows:

$$E(f) = E_{data}(f) + E_{smooth}(f)$$

Where E_{data} tells the cost of matching planes and E_{smooth} penalizes discontinuities ensuring the smoothness of the disparity map.

$$E_{data}(f) = \sum_{S \in R} C(S, f(S))$$

where C is:

$$C(S,P) = \sum_{(x,y) \in S-O} c(x,y,d)\, e^{1-\frac{S}{n}}$$

where P is the disparity plane, S is the current segment and O is the occluded portion of S.

$$E_{smooth}(f) = \sum_{(S,S')} u_{S,S'} \cdot \delta\big(f(S) \neq f(S')\big)$$

Where S and S' are neighboring segments. $u_{S,S'}$ is proportional to the common border length between S and S'. $\delta\big(f(S) \neq f(S')\big)$ has value of 1 if $f(S) \neq f(S')$, otherwise 0.

The pseudo code of the algorithm is presented as follows (Hong and Chen 2004):

```
1. Start with an initial labeling f.
2. Set success := 0
3. Select in a random (or fixed) order a disparity plane $P \in D$
    a. Find $\hat{f} = argmin\ E(f')\ among\ f'\ with\ one\ \alpha\ expansion\ of\ f$
    b. If $E(\hat{f}) \leq E(f)$, set $f := \hat{f}$ and success := 1
4. If success == 1 goto 2
5. Return f
```

For our experiments we used the implementation available for Open CV (OpenCV 2011).

We also implemented a custom disparity map reconstruction algorithm based on the implementation that (Beeler, et al. 2010) did for capturing facial geometry. First, we started calculating the disparity map by using a cross-correlation based algorithm, that would be used as input to a pyramidal approach of which results on the lower resolutions serve as input guide on the window size for the following higher resolution level. At each level of this Gaussian pyramid are computed at the pixel level so we can have dense matches on our binocular-stereo images. Then the information about the Euclidean distance among features on the horizontal axis are used as a depth map in order to reconstruct the geometry by disturbing the z axis position of the vertex of a two dimensional mesh geometry. This mesh is then refined by introducing photo-consistency and smoothing terms.

The first step is to generate the Gaussian pyramid. We do this by convolving the original image with a Gaussian function (also called Gaussian average or Gaussian blur) to later be scaled down. This effectively allows the lower resolution image to contain a local average that corresponds to a pixel neighbor on a lower level of the pyramid. Having a pyramid scaling factor of two produces the best results using different kinds of cameras which are often found on movies or artistic renderings due to the Bayer-pattern. Also subsampling helps to reduce noise. The lowest resolution allowed for the pyramid in our system was 128 pixels on the horizontal axis and 128 pixels along the vertical axis. If one of the axis dropped bellow that number then the immediate superior size is considered the minimum allowable size.

Matching starts at the lower resolution of the pyramid. At this level we used a three by three window to calculate the normalized cross correlation to implement a winner take al block-matching algorithm. Then we do the search for matching features in the epipolar line, that given the previously calculated image rectification, it is only done on the horizontal axis. The pixel p in image I is matched with all the pixels on the corresponding row on image J, within a given area. When the Euclidean distance on the luminance space on a pixel q on image J is minimum it's retained. Then the disparity at p is computed to sub-pixel accuracy by computing the normalized cross correlation values of p and q between its two neighbors in image J. We know that these two images have a epipolar line parallel to the horizontal axis, and that they can only be either on the left or the right to each other. The search for a matching feature has only to be done in the $\overrightarrow{IJ}$ direction.

Matching is done twice per level. The initial matching computes temporal matches for all pixels by using the disparity that estimates of the previous layer. Next, we make sure that the smoothness, uniqueness and ordering constrains are enforced. If it happens that a pixel doesn't comply with this constrains, then it is rematched by using the disparity estimates of its neighboring pixels that did comply with the constrains. This constraints simply accept or reject a pixel to the set of matched pixels along the binocular-stereo image.

The smoothness constraint is enforced by accepting only those pixels where more than half of their neighbors in a 3x3 window differ by a disparity of less than one pixel. The uniqueness constraint on the other hand is enforced by making sure about the bijectivity of the matched pixels. This means that if a pixel p on image I is matched with a pixel q on image J, then pixel q should be a match to pixel p. The ordering constrain is enforced by making sure that disparity at pixel p does not exceed the disparity of its right neighbor pixel by more than one pixel.

Because of the large amount of noise presented in the disparity map, some refinement needs to be done. Refinement for this algorithm refers to the linear combination of a photometric consistency term and a surface consistency term. The photometric consistency term favors solutions with normalized cross correlation. On the other hand, the surface consistency term favors smooth solutions. (Beeler, et al. 2010) recommends to have these two parameters exposed to the user so fine tunes can be made if needed. These terms are driven by a user specified smoothness parameter and a data driven parameter which ensures that the photometric term has the biggest weight on regions with good feature localization.

The disparity map refinement is done with sub pixel accuracy and it is updated at each iteration. As mentioned before the refinement is a linear combination of an adjustment in the direction of improved photometric consistency (d_p) and an adjustment in the direction of improved surface consistency (d_s). Where:

$$d_p = f(x) = \begin{cases} p - q - 0.5, & \xi - 1 < \xi_0, \xi + 1 \\ p - q + 0.5 \dfrac{(\xi - 1 - \xi + 1)}{\xi - 1 + \xi + 1 - 2\xi_0}, & \xi_0 \geq \xi - 1, \xi \\ p - q + 0.5, & \xi + 1 < \xi - 1, \xi_0 \end{cases}$$

In which given a pixel p on image I and its match on image J, we need to calculate the normalized cross correlation of p with q, q and q+1. We use the

$\overline{NCC} = \frac{1-NCC}{2}$ where 0 represents no error and 1 a wrong match. ξ_{-1}, ξ_0 and ξ_{+1} refers to the normalized cross correlation of pixels to the left, current and to the right respectively.

$$d_s = \frac{w_x\left(d_{x-1,\ y} + d_{x+1,\ y}\right) + w_y\left(d_{x,\ y-1} + d_{x,\ y+1}\right)}{2\left(w_x + w_y\right)}$$

Where w_x reduces smoothing across depth discontinuities.

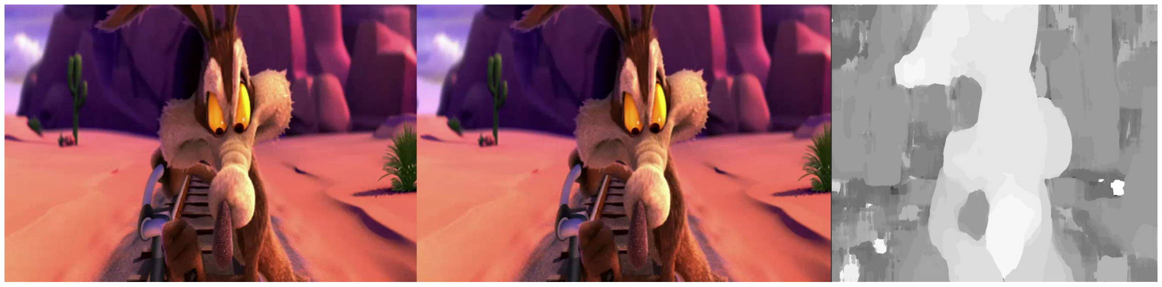


Figure 16. Examples of the binocular stereo image used as an input and the left disparity map obtained after running our algorithm.

Chapter 9: Detail Mapping

Our system takes a depth map as input in order to approximate scene geometry by using a surface mesh. As we discussed earlier, this depth map can come either from a disparity map from a binocular stereo image or from a rendering system. In order to maintain simplicity on the system and to make it simple to implement and replicate we decided to save this disparity map in any bitmap format currently available like the Microsoft Windows Device Independent Bitmap (BMP), Joint Photographic Experts Group (JPG or JPEG), Portable Pixel Map (PPM), TrueVision TARGA (TGA) or Portable Network Graphics (PNG). In order to avoid any noise or artifacts due to image compression we recommend using only lossless data compression formats or uncompressed formats. For this work we employed only BMP, PNG or PPM files. The use of different formats was totally transparent to our system since we relied on the DirectX 11 texture library that can automatically read many different formats and convert them to DirectX texture format. If the provided texture format is not compatible with the current DirectX implementation, then the system must provide an implementation to a format understandable to Direct X or a direct conversion to the DirectX format.

Most of the bitmap formats provide a fixed depth resolution meaning that they can only represent a finite number of colors for each pixel. This is a problem, since our system is taking this map as a depth map that represents the distance from each pixel to the object that is being represented at this position.

Distance is a continuous numerical measurement of which resolution is infinite. If we try to map the numerical distance values into a finite depth resolution format, it can lead to quantization errors on the haptic rendering, which needs to be controlled in order to produced pleasing images and results.

We call detail mapping to customization of the tone-mapping algorithm specially used so that details encoded on the depth map can be rendered on the haptic device. Tone mapping is a technique often used on image processing and computer graphics to map a high dynamic range image into a lower range image; in this way lower range image contains compressed information without loosing details from the high dynamic range image. Essentially tone mapping algorithms deal with the problem of strong contrast radiance values so that they can be used or displayed on lower range systems, while preserving scene detail and color appearance that closely matches to the high dynamic range image.

There are mainly 2 different kinds of tone mapping algorithms:

Global. These are non-linear functions based on the overall image luminance and some other variables. First, we must find a function that is optimal according to a particular image and every single image is mapped exactly the same way, totally independent from the value of surrounding pixels of the image. This kinds of algorithms are typically very fast; nonetheless their main drawback is the loss of contrast.

Local. The parameters of the non-linear function changes at every pixel depending on its surrounding neighbors. In other words, the parameters change according to the locality. These algorithms are slower than the global ones and they are very prone to artifacts like halos and ringing; generally their results seem exaggerated. On the other hand, they are very good while trying to have control of local contrast.

The simplest and most naïve tone-mapping algorithm is done by normalizing pixel values:

$$L = \frac{Y}{Y+1}$$

This function maps any radiance values (Y) from $[0, \infty)$ to a displayable output range of $[0, 1)$.

Since the main goal on tone mapping algorithms is to generate a perceptually equivalent low range image from a high dynamic range image, few of the tone mapping algorithms exploits properties of the human visual system. For example, human eyes cannot perceive the actual intensity for every single part of the scene, but it can understand the local intensity variations in different parts of the scene. (Durand and Dorsey 2002) exploited this phenomenon by suppressing global intensity variations without affecting appearance.

In this work we decided to implement a couple of different algorithms to analyze their performance and results when applied to high dynamic range depth maps. The first algorithm we tried in the experiments for this work is the local adaptation model proposed by (Ashikhmin 2002) that also exploits the way human vision system perceives images. Tone mapping is implemented by using a linear mapping that is applied locally to the different regions of an image. There the contrast does not varies by a big amount, this preventing the formation of halos or very dark edges. This is done in two simple steps:

1.- Estimate the local adaptation level and contrast. The image is firstly segmented into different regions where the contrast does not vary significantly. This is done by using the Gaussian average over a small window of neighbors surrounding a pixel. Local contrast is defined as:

$$c(x,y) = \frac{L(x,y)}{L_a(x,y)} - 1$$

Where L is the pixel luminance and L_a is the local adaptation level which is the Gaussian average over the pixel (x,y).

2.- Apply the tone mapping function. One of the main goals of this paper was to have locally linear mapping algorithm that preserves details throughout the image. Once we have $L_a(x,y)$ we can apply a tone mapping function ($TM(L)$) which will create a tone mapped image $TM(L_a(x,y))$. The main purpose of this function is to compress the high dynamic range image while trying to preserve the overall perception of brightness. By using the formula from stage 1 we can have the function for the final mapping:

$$L_d(x,y) = \frac{L(x,y)TM(L_a(x,y))}{L_a(x,y)}$$

Since the real image can have very high local contrast and we are treating it as a Gaussian average over a constant size window it will lead to halos (often called inverse gradients). (Ashikhmin 2002) deals with this by dynamically adjusting the size of the region used to compute the adaptation luminance, so that in smooth regions the size is larger than in regions with high level of detail, where the windows size often shrinks up to only one pixel.

Another tone mapping algorithm we choose to explore in this work is histogram equalization for tone mapping. This method is often used to alter the global contrast of images; thus through this adjustment the intensities can be better distributed through the histogram. This is effectively done by spreading out the most frequent intensity values throughout the whole range of the histogram. This algorithm is very easy to implement, inexpensive to compute and is invertible. If the histogram equalization function is known, then the original histogram can be recovered. The main drawback of this algorithm is that it can produce undesirable noise while decreasing the usable signal. Due to the global nature of this algorithm if parameters are not correct, it also can cause halos and darker edges.

In our case, we apply this operator over the depth map that is a gray scale image $\{x\}$. Let n_i be the number of times the gray level i appears. Then the probability of occurrence of a pixel at level i in the image is:

$$p_x(i) = p(x = i) = \frac{n_i}{n}, 0 \leq i < L$$

Where L is the total number of gray levels on the whole image, n is the total number of pixels on the image, and $p_x(i)$ the histogram value for pixel (x,y), normalized to [0,1].

We also must define the cumulative distribution function of pixel (x,y) as:

$$cdf_x(i) = \sum_{j=0}^{i} p_x(j)$$

If we apply a transformation of the form $y = T(x)$, we can produce a new image y so that its cumulative distribution function is linearized across the value range for a value K:

$$cdf_y(i) = iK$$

Using the properties of the cumulative distribution function we can state that:

$$y = T(x) = cdf_x(x)$$

Where T goes from [0,1]. If we want to map back the values to their original value we can use the following function:

$$y' = y \cdot (\max\{x\} - \min\{x\}) + \min\{x\}$$

On our experiments we were able to essentially preserve most of the detail on the limited resolution haptic device.

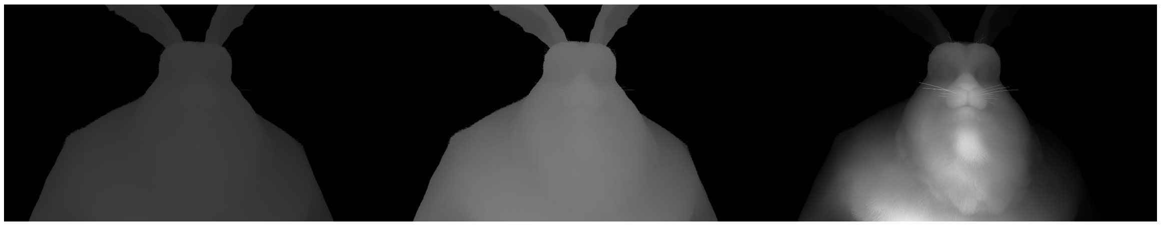


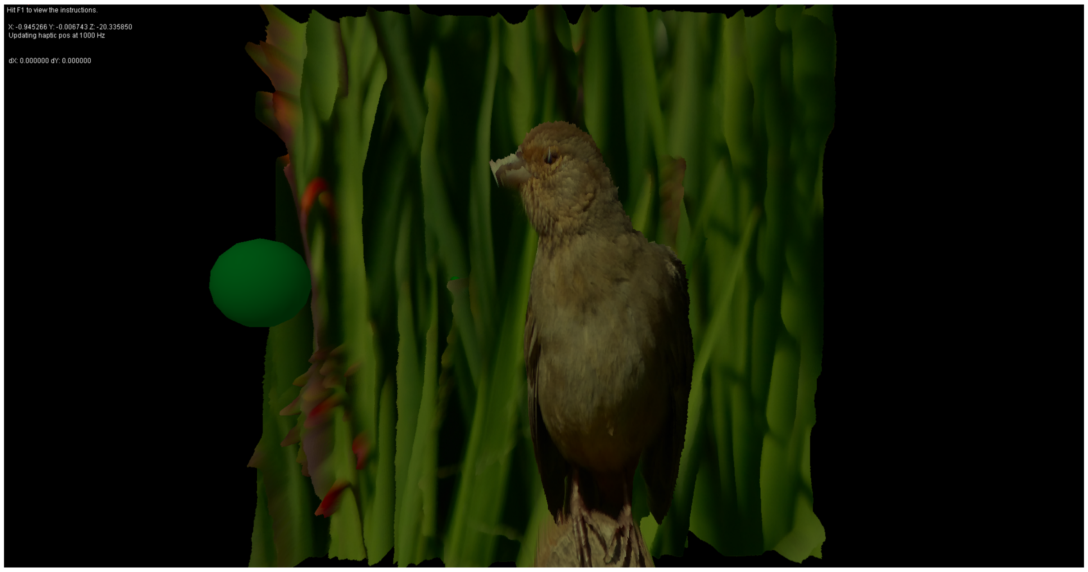
Figure 17. Detail mapping results.

Chapter 10: Rendering from depth map

Once the depth map is generated either from a rendering system or from a binocular-stereo disparity map the algorithm to render it is very straight forward. This algorithm is very obvious and simple. It doesn’t represent any complication while implementing it in most of the cases.

The very first thing to do is construct a bidimensional mesh of vertices over a 3D space, it is recommended that this mesh is axis aligned so that further calculations are simplified avoiding the need to either undo transformations on the mesh or to apply transformations to further calculations. It is also recommended to implement this mesh in a memory coherent data structure, such as a simple struct array, so that we can take advantage of caching mechanisms implemented on General Processing Units and Graphics Processing Units.

It is also important to remember that rendering systems have conventions in the way they represent axis and geometry on screen. The most common schemas are left or right handed. Right handed rendering systems like Open GL, represent the X axis pointing to the right, Y axis pointing up while the negative Z axis points forward. Rotations are counterclockwise around the axis of rotation. Left handed rendering systems like Render Man represent the X axis point to the right, Y axis pointing up and the positive Z axis pointing forward. Rotations are always clockwise around their axis.


```
Hit F1 to view the instructions.

X: -0.945266 Y: -0.006743 Z: -20.335850
Updating haptic pos at 1000 Hz

dX: 0.000000 dY: 0.000000
```

Figure 18. When going with a very high resolution depth map some undesirable effects can appear and can slow down the frame rate.

It is important to state that the number of vertex to render depends mostly on the scene content and the desired reproduction quality. In our experiments we found that not all the times going the highest definition available is the best choice. Sometimes high frequency details like fur and hair disrupt the haptic feeling and produce visual artifacts on screen.

Instead, it is wise to experiment with different resolutions that provide pleasing haptic and visual sensations, while keeping the signal fidelity at a desirable level. In order to achieve the resolution flexibility that allows us to experiment with the different the resolution parameters we implemented both the bilinear interpolation and the bicubic interpolation algorithms to resample the original depth map image.

The bilinear interpolation algorithm produces very good results having a small memory and processing footprint, although the bicubic algorithm produces surfaces smoother that the bilinear algorithm it requires a bigger processing footprint. Since the resulting mesh could be saved in a file to be later retrieved for our application we recommend using the bicubic algorithm to produce better quality

The bilinear interpolation algorithm works bidimensional grid, by performing linear interpolation for each axis, one at a time. Although interpolation at each axis is linear, the interpolation as a whole represents a quadratic function at the sample position. Suppose that we want to find the value of the function at a point $P = (x, y)$ where we know the value of this function at the points $Q_{11} = (x_1, y_1)$, $Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1)$ and $Q_{22} = (x_2, y_2)$.

If we do the interpolation on the X axis:

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \; where \; R_1 = (x, y_1)$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \; where \; R_2 = (x, y_2)$$

When doing the interpolation in the Y axis:

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2)$$

If we combine this functions, we can find the estimate of $f(x, y)$ by:

$$f(x,y) \approx \frac{f(Q_{11})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y_2-y) + \frac{f(Q_{21})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y_2-y) + \frac{f(Q_{12})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y-y_1) + \frac{f(Q_{22})}{(x_2-x_1)(y_2-y_1)}(x-x_1)$$

Different than the bilinear interpolation algorithm, whic takes only the four surrounding samples near the sampling point, the bicubic takes the 16 samples surrounding the sampling point in a four by four grid. Suppose that the function values f and the derivatives f_x, f_y and f_{xy} are known at the four corners of a unit square. The interpolated surface can be written like:

$$p(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j$$

The problem comes when determining the sixteen coefficients of a_{ij}. Matching $p(□, y)$ with the function give us the following equations:

$$f(0,0) = p(0,0) = a_{00}$$

$$f(1,0) = p(1,0) = a_{00} + a_{10} + a_{20} + a_{30}$$

$$f(0,1) = p(0,1) = a_{00} + a_{01} + a_{02} + a_{03}$$

$$f(1,1) = p(1,1) = \sum\nolimits_{i=0}^{3}\sum\nolimits_{j=0}^{3} a_{ij}$$

And for the derivatives on the X and Y axis:

$$f_x(0,0) = p_x(0,0) = a_{10}$$

$$f_x(1,0) = p_x(1,0) = a_{10} + 2a_{20} + 3a_{30}$$

$$f_x(0,1) = p_x(0,1) = a_{10} + a_{11} + a_{12} + a_{13}$$

$$f_x(1,1) = p_x(1,1) = \sum_{i=1}^{3} \sum_{j=0}^{3} a_{ij} i$$

$$f_y(0,0) = p_y(0,0) = a_{01}$$

$$f_y(1,0) = p_y(1,0) = a_{01} + a_{11} + a_{21} + a_{31}$$

$$f_y(0,1) = p_y(0,1) = a_{01} + 2a_{02} + 3a_{03}$$

$$f_y(1,1) = p_y(1,1) = \sum_{i=0}^{3} \sum_{j=1}^{3} a_{ij} j$$

And the equations for the derivative of xy

$$f_{xy}(0,0) = p_{xy}(0,0) = a_{11}$$

$$f_{xy}(1,0) = p_{xy}(1,0) = a_{11} + 2a_{21} + 3a_{31}$$

$$f_{xy}(0,1) = p_{xy}(0,1) = a_{11} + 2a_{12} + 3a_{13}$$

$$f_{xy}(1,1) = p_{xy}(1,1) = \sum_{i=1}^{3} \sum_{j=1}^{3} a_{ij} ij$$

This problem can be formulated as linear system of the form $A\alpha = x$ where the unknown parameters α_{ij} are expressed as:

$$\alpha = [a_{00}\ a_{10}\ a_{20}\ a_{30}\ a_{01}\ a_{11}\ a_{21}\ a_{31}\ a_{02}\ a_{12}\ a_{22}\ a_{32}\ a_{03}\ a_{13}\ a_{23}\ a_{33}\ ]^T$$

and

$$x = \left[f(0,0)\; f(1,0)\; f(0,1)\; f(1,1)\; f_x(0,0)\; f_x(1,0)\; f_x(0,1)\; f_x(1,1)\; f_y(0,0)\; f_y(1,0)\; f_y(0,1)\; f_y(1,1)\; f_{xy}(0,0)\; f_{xy}(1,0)\; f_{xy}(0,1)\; f_{xy}(1,1)\right]^T$$

Once the depth values are resampled we can calculate the corresponding depth values for each vertex. Most of the times this calculation will lead to almost unperceivable changes on the bidimensional plane. We must have a constant multiplier for each vertex so that we can scale the perturbation on the 2D surface until we get a pleasant result.

Some times the most important features of the scene lay between certain depth. We can apply certain depth thresholds in order to give more protagonism to certain scene elements.

When testing our experiments we realized that extremely concave surfaces lead to vibration problems on the haptic device. To solve this problem, we introduced a smoothing term based on the surface derivatives to avoid it. We will talk more about this on chapter 12 of this thesis.

Once the surface has been calculated we must apply the color texture to it. This is a very simple task and most of the rendering systems provide capabilities to do this. It is important to mention that we must select a shader on the rendering system where none of the diffuse or specular components are present, only color derived from the texture images should be rendered.

Chapter 11: Head-Tracking

In order to complete all of the depth cues needed to implement the fish tank virtual reality we must implement a mechanism in order to track the users head position. We came with many different solutions that varied in many different aspects such as hardware availability, cost, responsiveness and accuracy. There are professionally available optical systems that track markers in order to capture the 3D position of an element on the scene. This usually rank number one on responsiveness and accuracy among all of the other systems. However it has many drawbacks that does not align with the goals of our system. The cost at the moment of writing this thesis was more than twenty thousand dollars, which was prohibitive for the experiments for this thesis since our goal was to have a sub one thousand dollar system. These systems also require the use of facilities in which a collection of at least 6 different cameras can be permanently installed and constant calibration. Additionally there is the need of wear markers every time the user needs to use the motion capture system. We also tried different facial recognition algorithms, these algorithms worked well when certain conditions were meet, but produced poor results when dealing with adverse situations such as poor illumination or with the use of the 3D glasses.

For this thesis we decided to use the Microsoft Kinect (Microsoft Corporation 2011) depth camera and its publicly available software development kit. The simplicity of its application programming interface, its commercial availability, its low cost and its good performance on most of the conditions made it the perfect candidate for our experiments.

The Microsoft Kinect motion capture system can be divided into two elemental parts: the hardware, which captures a depth image; and the software, which uses a real time probabilistic algorithm that labels the different body parts with per frame initialization. Segmentation is treated as a per pixel classification task; in this way a combinatorial search is avoided and since each pixel is totally independent, the classifier can be run on parallel for each pixel.

For training data several realistic depth images of humans in many shapes and positions were generated from a motion capture database. This data was used to train a deep randomized decision forest classifier, which avoids over fitting by using a very large amount of training images that ranges on more than five hundred thousand images.3D translation invariance is achieved by doing a discriminative depth comparison. Finally, spatial modes of the inferred per-pixel distributions are computed using mean shift resulting in the 3D joint proposals.

In order to classify the samples this work employs a simple depth comparison for the features. At any given pixel (x) the features are computed as following:

$$f_\theta(I,x) = d_I\left(x + \frac{u}{d_I(x)}\right) - \left(x + \frac{v}{d_I(x)}\right)$$

Where $d_I(x)$ is the depth at pixel x in the image I, and u and v describe the offset in a normalized space, which is calculated as $\frac{1}{d_I(x)}$ and ensures its depth invariance. If the pixel lies in the background or it is outside certain threshold, the depth probe $d_I(x')$ is given a large positive constant value. If considered individually these features give a week signal of which part of the body they belong, however if we take into account the decision forest they provide a stronger indication of their correct labeling.

A decision forest is a collection of T decision trees, each one consisting of split and leaf nodes. Each one of the split nodes consists of a feature f_θ and a threshold τ. To classify the pixel x in image I, we must start at the root and evaluate the feature function, which will branch to the left or to the right according to the threshold τ. At the leaf node reached in tree t, the learned distribution $P_t(c|I,x)$ over the selected body part label c is stored. The distribution is the averaged for all the trees in the forest to give a final classification:

$$P(c|I,x) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|I,x)$$

Each tree is trained in a different set of images. Each tree is trained using the following algorithm (Leptit, Lagger and Fua 2005):

1.- Randomly propose a set of splitting candidates $\phi = (\theta, \tau)$ (feature parameters θ and thresholds τ).

2.- Partition the set of examples $Q = \{(I,x)\}$ into left and right subsets by each ϕ:

$$Q_1(\phi) = \{(I,x) \mid f_\theta(I,x) < \tau\}$$

$$Q_r(\phi) = \frac{Q}{Q_1}\phi$$

3.- Compute ϕ giving the largest gain in information:

$$\phi^* = \underset{\phi}{argmax}\, G(\phi)$$

$$G(\phi) = H(Q) - \sum_{s\in\{l,r\}} \frac{|Q_s(\phi)|}{|Q|} H(Q_s(\phi))$$

where Shannon entropy $H(Q)$ is computed on the normalized histogram of body part labels $l_I(x)$ for all $(I,x)\epsilon Q$.

4.- If the largest gain $G(\phi^*)$ is sufficient, and the depth in the tree is below a maximum, then recurse for the left and right subsets $Q_l(\phi^*)$ and $Q_r(\phi^*)$.

Once all the samples have been labeled we need to find the final positions of the joints. A simple way to do it would be to find the global 3D centers of probability mass for each part. However, noisy samples could degrade the quality of the result. Instead a local mode-finding approach based on the mean shift using a Gaussian kernel is employed, where the density estimator per body part is:

$$f_c(\hat{x}) \propto \sum_{i=1}^{N} w_{ic} exp\left(-\left\|\frac{\hat{x}-\hat{x}_i}{b_c}\right\|^2\right)$$

Where $\hat{x}$ is a coordinate in the 3D space, N is the number of pixels, w_{ic} is a pixel weighting, $\hat{x}_i$ is the reprojection of the image pixel x_i into world space given depth $d_I(x_i)$, and b_c is the learned per part bandwidth. The pixel weighting can be calculated as:

$$w_{ic} = P(c|I, x_i) \cdot d_I(x_i)^2$$

Once the user skeleton has been identified and labeled, the Kinect SDK simply delivers it, at frequency of 30 frames per second in average, in an array with the world coordinates for each joint. In our system we used the coordinates of the users head as the world position for the camera, aiming at the center of the mesh. We found that even thought this method works on most of the cases, there are certain limitation inherited directly from the hardware, like when the users figure gets clipped by the camera field of view, when some occlusions appear, fast movements or when the user is very close to the camera. We overcame these drawbacks with a simple two-step solution in order to provide a smooth signal for the head position:

First, we did a weighted average among the joints that contribute the most to the head position: base neck, left shoulder, right shoulder and the head it self.

Second, we did a Gaussian weighted average in the time domain over the weighted position gotten from the previous step.

Some times, while testing the system, certain individuals were particularly sensitive to the camera movement on screen. We decided to parameterize the head movement by simply having a multiplier for each of the axis for the head smoothed 3D coordinate.

Chapter 12: Haptics

In our system we decided to use the Novint Falcon Gaming system. We decided to use this system because of its great commercial availability, low entrance cost and the big developer community around the product. This will ensure that any person who would like to replicate our system could do it with any major problems. The Novint Falcon was the first consumer oriented 3D Haptic Device that allowed people to virtually touch elements in a scene. The Novint Falcon consists of 3 arms connected to a conical body that can independently apply forces in different directions so that the user can feel them and interpret its feedback as well as the many different sensations, ranging from touching an object to feeling inertial forces to the kickback of a pistol. This hardware is very precise and can sense movements with submilimiter accuracy.

Historically Novint has focused more on the software aspects of haptics rather than on the hardware side. Novint in its Novint Falcon SDK provides a low level driver called HDAL, which stands for Haptic Device Abstraction Layer. This driver handles the low level communications between the Falcon microprocessor and the computer. There is another high level framework that lies on top of the HDAL, named HFX, that is a simple and easy to use set of rules to create reaction forces and that is mainly used by videogame studios.

For our system we decided to use the Haptic Device Abstraction Layer framework (HDAL), mainly because it provides low level access to the hardware in order to implement more advanced features. We used this framework by using Microsoft Visual Studio 2010 on C++ and Microsoft Direct X 11.

It is very important to discus the limitations of the sensations that a haptic device can produce and those sensations that we can feel in the real world. First of all, in the real life we feel object using complete regions of our skin. This allows us to evaluate the surface in many parts object at the same time. In just one moment we can evaluate the surface texture and the derivatives of the object geometry. Meanwhile on the haptic device, since we are sensing just through a handle, we must change the position over the time in order to sense object texture or any other geometry derivative.

Another big difference is the sampling rate. in the real world there is an infinite sampling rate since geometry, force, distance and time are continuous. This means that given any two different times, it is possible to find a third value between the two. This also holds for force, distance and geometry position. In contrast in the computers world this is different. Given processing power an memory limitations, the length and sample rates are discrete. This is, that it is possible to have two values that are so close together that the computer might think that they represent the same value. In order to overcome this, we must have a higher resolution-sampling rate, so that collisions between values are rare and give the impression to be continuous. Research have proved that in order to have an impression of touching a smooth surface the sampling rates must rely between the 500Hz and 1,000 Hz.

This numbers are very important, if we do the math we merely have 1 millisecond in order to do all the calculations needed and deliver the signal to the haptic device; that's why it provides touch feedback to the user. This represents a special problem if you have some other complicated calculations such as graphics and sound processing linked to your system. Calculations have to be done as fast as possible and in a clever way so that you can achieve the required sampling rate in order to provide a satisfying sensation to the final user.

For our system we implemented the spring mass damper model where:

$$F_s = -kx$$

And a damping force of:

$$F_d = -cv = -c\frac{dx}{dt} = -c\dot{x}$$

Where F_s is the force, k is the spring constant and x is the position of the object, c is the damping coefficient. If we treat the mass as a free body and we apply the Newton's second law, the total force applied on the object is:

$$F_T = ma = m\frac{d^2x}{dt^2} = m\ddot{x}$$

Where a is the acceleration of the mass and x is the displacement of the mass relative to a fixed reference. The HDAL API comes very handy to all types of programmers. It abstracts most of the low level interaction with the Novint Falcon but stills gives the programmer control of every single step in the haptics pipeline. The HDAL is implemented as a series of layers:

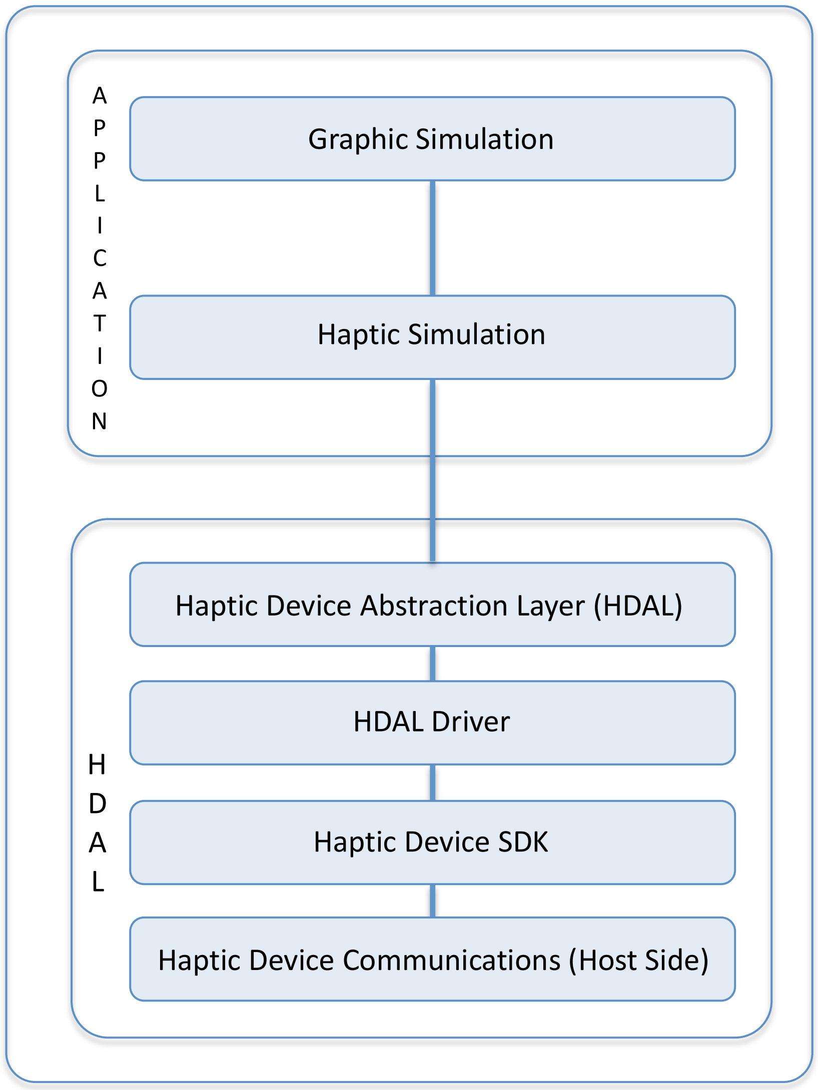


Figure 19. HDAL Layered Architecture (Novint Technologies Incorporated 2008).

On the application Layer we can find the two main components that interface with the final user experience: the graphics simulation layer and the haptics simulation layer. The graphics simulation layer is in charge of rendering the elements on the screen, while the haptics layer is in charge of calculating the forces that the haptic device will produce and the actions that it would trigger on the scene. The main important task of the programmer is to have these two layers in synchronization, so that the visual feedback the graphics layer provides is coherent with the haptic feedback derived from the haptics simulation layer.

The haptics simulation component communicates directly to the Haptic Device Abstraction Layer (HDAL) through the HDAL API. The main task of the HDAL layer is to provide communication with the haptic device so that we can set forces to the device and get feedback from it in order to, for example, find the position of the grip. This communication is ideally done with a frequency of one kilohertz, so that every one thousandth of a second the user can read the servo position, calculate the appropriate forces depending of the interaction on the application and set these forces on the device.

The HDAL driver layer manages a family of drivers that provides an abstraction for the communication between the computer and the haptic device itself. It basically translates commands given to it on high-level languages to a set of commands that the specific Haptics Device SDK can understand. The Haptic Device SDK layer is a low level, device specific to set of commands that provide an interface to the HDAL drivers in order to communicate agnostically with the haptic device. The Haptic Device Communications layer deals with internal communications and calculations done entirely on the host side. For example, it retrieves the position for each arm on the Novint Falcon and provide an access point to this date so that it can be used to calculate the approximate position of the grip.

In this chapter we will concentrate on describing the algorithm that controls the haptics simulation layer. For the collision detection algorithm, on our first experiments we decided to use a very basic and naïve algorithm in our first experiments. Throughout this algorithm we merely project back the position of the cursor on the screen over the Z axis, perturbed it accordingly to the depth map position and did the proper spring damping calculations by using the normal direction on the contact surface position in order to find the correct force to apply to the Novint Falcon handles.

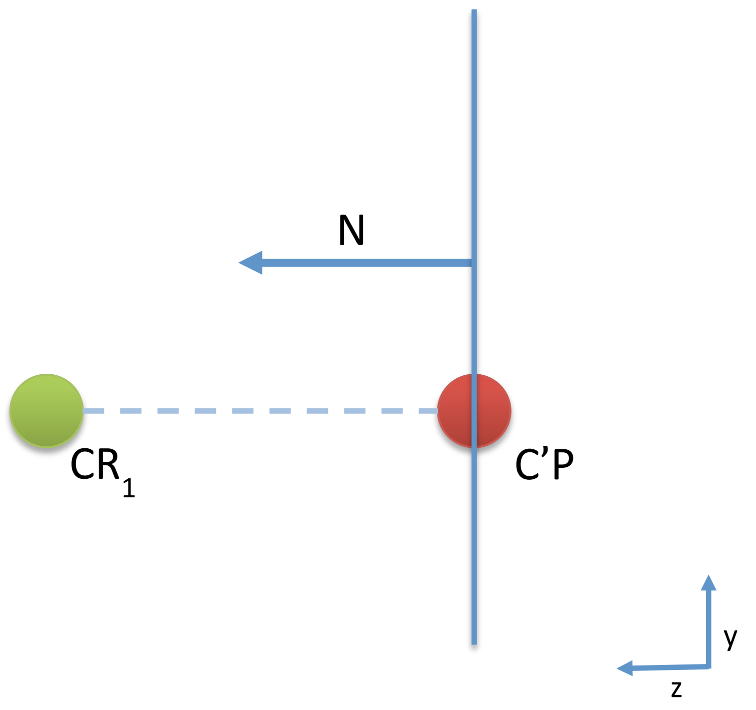


Figure 20. Naïve force damping model implemented using the surface normal.

Where N is the surface normal, C is the original cursor position, C' is the projected cursor position. This simple algorithm works very well on convex surfaces and due to its simplicity it could be an ideal candidate for our implementation. Nonetheless it provides unstable results when concave surfaces are present. If we leave the force calculation as simply getting the proper forces for each given period of time we can run into the problem on which the consecutive previous force conflicts with the current calculated force creating an instable force that leads in to handle vibration.

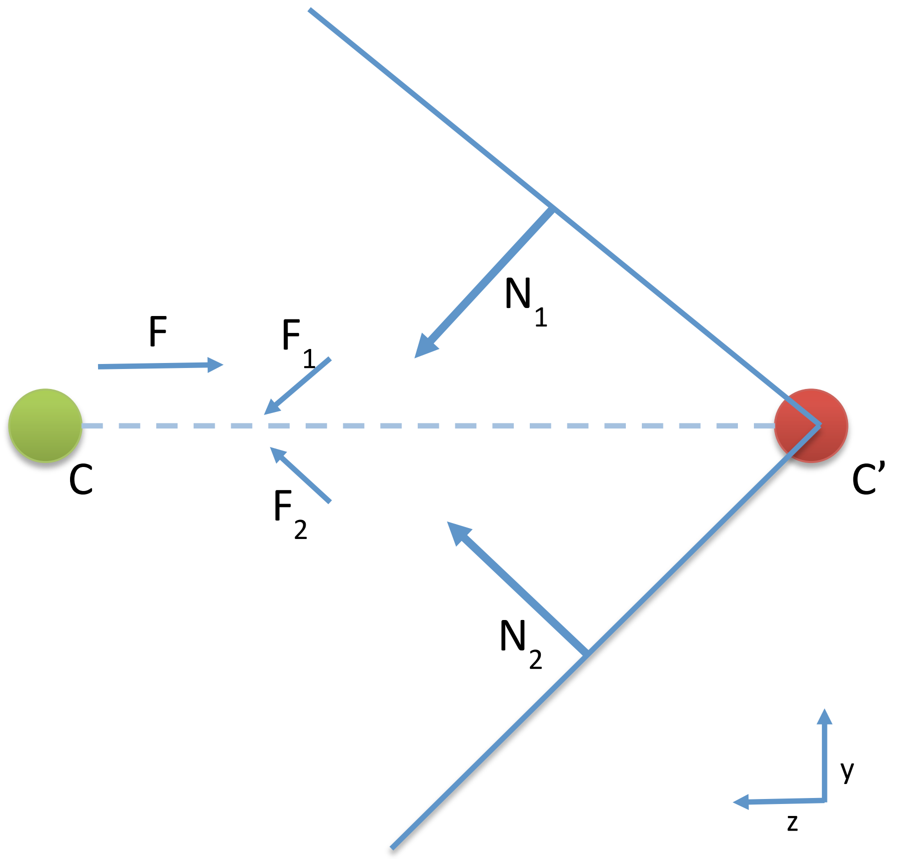


Figure 21. Diagram illustrating an example on which the naïve force calculation can produce vibration on the haptic device handle.

To solve this problem we simply proposed an algorithm in which we calculate the position of the projected cursor as the weighted average of the position of a set of four surrounding cursors with a certain offset from the original one as show on figure 7. This effectively smooth out the surface reducing the convexity of the mesh, while maintaining the overall visual appearance that the cursor movement is coherent with the mesh topology.

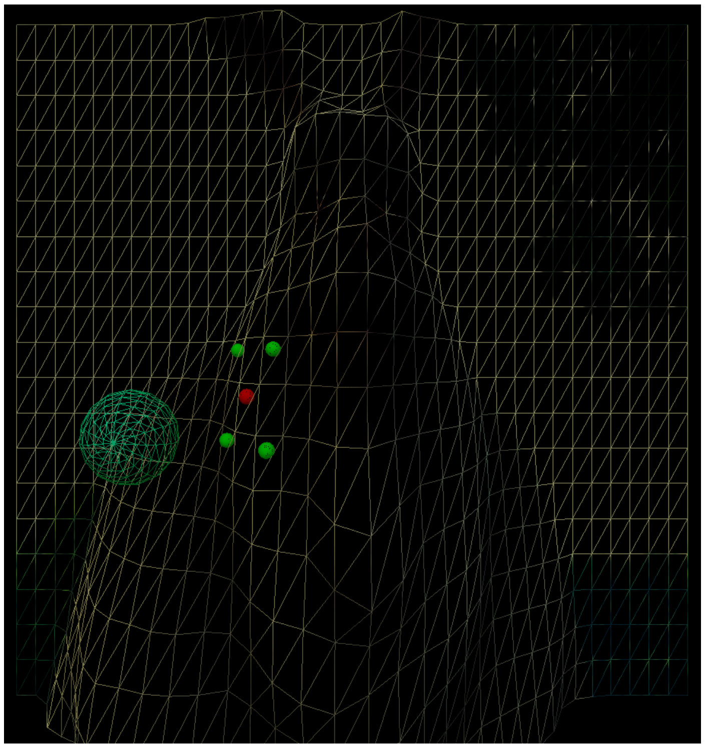


Figure 22. Projected cursor position calculation.

Where we have a set of 4 points that have an equal offset on the X, Y, -X and –Y axis:

$$P = \{P_1, P_2, P_3, P_4\}$$

And the point where the cursor was originally projected on to the mesh C_1. The weight of each point on set is equal, and it is exposed as a parameter along with the weight for the point C_1. We use the following equation to calculate the final position of the projected cursor on the mesh:

$$\bar{c} = \frac{\sum_{i=1}^{n} P_n w}{\sum_{i=1}^{n} w} + \frac{c_1}{w_{c_1}}$$

We put special attention to the refresh rates for the force calculation. At the very beginning we had some grip vibrating problems due to an incorrect refresh rate for the haptic because we were calculating the forces every for every frame. Then, we noticed this fundamental problem and change our code so that the forces were calculated on a different thread allowing us to have refresh rates very close to the 1kHz (997 Hz on average), which produced a very smooth sensation while virtually touching the scene.

On informal user experiments in our laboratory, users were remarkably exited when using our system. For the great majority of them, it was the very first time in their lives, when they were watching a 3D image on stereoscopic display, which was perspective corrected according to their head position along with the ability to touch elements that they perceived on the display. We are very excited on the possibilities that a system like this opens on the user immersion spectrum.

This is not only a system that completes depth cues and provide a better experience, this is also a system that is very easy to implement with a remarkable low entry cost.

Chapter 13: Results

Throughout this thesis we have discussed the parts that integrate this project. We have also discussed some of our decision on which algorithms to use and some of the drawbacks they could have. In order to evaluate the results of our system we decided to do an informal study measuring 2 quantitative aspects:

Average fixing time. We asked the users to touch a set of spheres randomly located on a 3D scene and measure the time it took them to actually touch all the spheres in the scene.

Average distance. We asked the users to touch a set of spheres randomly located on a 3D scene and indicate when they feel like they were touching the sphere. We measure the average distance from the point where the users indicated they were touching the spheres and the actual position of the sphere.

We did this measurements using the same hardware that we used to build this system but varying the input methods and hence sometimes blocking some of the functionality that gives and advantage to our system. The combinations we used are:

1.- Using the our system using the Novint falcon as merely a 3D input device and no haptic feedback.

2.- Using our system with no stereoscopic display. This way the user cannot use depth information from stereopsis as input.

3.- Using our system with no head tracking. This way the user does not get information from different depth cues like motion parallax, depth from motion or even occlusion.

4.- Using our complete system. This way the user gets all depth cues implemented in this work and haptic feedback.

The results are shown in the following table.

	1	**2**	**3**	**4**
Average Fixing Time	14.32 seconds	7.79 seconds	7.22 seconds	6.87 seconds
Standard Deviation	3.13	2.06	0.56	0.44

Figure 23. Table showing the average time it took to user to indicate that they were touching the target.

	1	**2**	**3**	**4**
Average Distance	3.69 units	0.7 units	0.72 units	0.71 units
Standard Deviation	1.35	0.36	0.11	0.33

Figure 24. Figure showing the average distance from which the user indicate they felt like they were touching the target to the actual position of the target.

The results clearly indicate an improvement when combining an stereoscopic display, head-tracking technologies and haptic feedback. We also found out that haptic feedback provided a greater improvement on average fixing time and average distance than completing the depth cues.

We also asked our set of experimental users to give some qualitative feedback about their level of immersion. We asked them to grade from 0 to 10 their level of immersion where 0 was the lowest level of immersion and 10 was the greatest. The results are show in the next table.

	1	**2**	**3**	**4**
Level of immersion	7.71	8.43	8.42	9.71
Standard Deviation	1.11	0.97	0.53	0.48

Figure 25. Qualitative evaluation of the level of immersion from the users.

We can see that user immersion was the greatest in this experiments when using the system that implements the stereoscopic display, head-tracking and haptic feedback the lowest when using the Novint Falcon without haptic feedback. This clearly shows that haptic interfaces are very important part of user immersion for our system.

We also measured the amount of time it took to the users to adapt to our system. The fastest learning time was on approach 1, the mapping of their hand movement to the cursor on screen was just natural to most of the users. The learning time significantly increased when using the Novint Falcon from 8.42 seconds on average to the approach 1 to 42.6 seconds on approach 5.

Chapter 14: Conclusion

Nowadays everybody can buy displays out of the shelf that provide very good resolution, and very high contrast ration, with amazing refreshing rates. However the way we perceive those images has not changed dramatically since its introduction on the early nineteenth century until recently. In the seventies and eighties the first explosion of 3D movies happened; this basically allowed the user to perceive depth in a picture that other wise would feel flat. At that time these 3D images were done by using anaglyphs, which effectively produced the desired 3D effect, nonetheless its fall short on reproducing the original colors of the image. On the last decade displays technology manufacturers all around the world have put a lot of effort to provide an effective way to display 3D images. In this way the flutter shutter and the polarized image method the most popular ones to provide an stereoscopic image to the final user.

Even thought the technology to display 3D images provides an outstanding quality, the way people can immerse in the images is still very limited. For example, watching a 3D movie on the cinema, where a person can effectively perceive depth of the frames that are being projected on the cinema display, but the immersion effect gets totally broken when the user changes its viewpoint position and the image on the screen keeps the same perspective from the viewpoint it was recorded from.

Throughout this work we introduced a new system that increases the user immersion by completing some of the missing depth cues and by providing the sense of touch to standard 3D images. That was strategically done so that the costs of replicating remain very low. In fact this specific system was built with less than one thousand dollars. This system can be easily implemented and enhanced by almost any programmer in the world. We built it by using exclusively publicly available application programming interfaces, software development kits and most of them were entirely free for academic and non-commercial use.

One of the main contributions of this system is the completion of the depth cues from a standard 3D image. For that we implemented different ways to recover the geometry either from a synthetic image while rendering or as a post process from a binocular image. Given that we recovered enough geometry from the scene, and we were able to create novel viewpoints from the scene. We exploited that by linking the user head position, using the Microsoft Kinect Hardware and SDK, to the rendering viewpoint of the system. With this we effectively implemented a perspective corrected viewpoint of the scene, and in some cases we even completed the movement occlusion depth cue.

We also implemented a haptic rendering algorithms so that the users have the ability to touch scene elements from 3D images. Since we had already recovered the geometry on previous steps the haptic rendering part perfectly fit this system. We used a mass spring damper algorithm along with some other custom algorithms in order to reduce concavity of surfaces to avoid artifacts while haptic rendering.

We also thought about enhancements to this system that were out of the main spectrum of this work. Such as the implementation of touch shaders, where each part of the scene geometry can have different hardness or stickiness parameters that can enhance user immersion. We also look forward to a robust and reliable framework to recover disparity maps from binocular image. In our survey we found a big number of algorithms with remarkable results that lacked of generality. Those algorithms that take into account general cases tended to produce results with only fair quality. Using algorithms like patch map reconstruction could help to improve the quality of the results of disparity map construction and scene completion.

References

Çavuşoluğlu, M., A. Sherman, and F. Tendick. “Design of Bilateral Teleoperation Controllers for Haptic Exploration and Telemanipulation of Soft Environments.” *IEEE Transactions on Robotics and Automation* 18, no. 4 (2002).

Arsenault, R., and C. Ware. “Eye-hand coordination with force feedback.” *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2000.

Ashikhmin, M. "A Tone Mapping Algorithm for High Contrast Images." Edited by P. Debevec and S. Gibson. *Eurographics Workshop on Rendering*, 2002.

Avila, R. S., and L. M. Sobierajski. “A haptic interacion method for volume visualization.” *Proceedings of IEEE Visualization*, 1996.

Avizzano, C. A., F. Bargagli, A. Frisoli, and M. Bergamasco. “The hand force feedback: analysis and control of a haptic device for the human-hand .” *IEEE International Conference on Systems, Man, and Cybernetics*, 2000.

Azarbayejani, A., T. Starner, B. Horowitz, and A. Pentland. “Visually controlled graphics.” *PAMI*, 1993.

Bach-Y-Rita, P., C. Collins, F.A. Saunders, B. White, and L. Scadden. “Vision Substitution by Tactile Image Projection.” *Nature*, 1969 8-March: 963-964.

Barnard, S. T. “Stochastic stereo matching over scale.” (IJCV), no. 3 (1989).

Bau, O., I. Poupyrev, A. Israr, and C. Harrison. “TeslaTouch: Electrovibration for Touch Surfaces.” *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, 2010.

Beeler, T., B. Bickel, P. Beardsley, R. Sumner, and M. Gross. "High-Quality Single-Shot Capture of Facial Geometry." *Proceedings of ACM SIGGRAPH* (ACM), July 2010.

Berthier, A. “Images stéréoscopiques de grand format.” (Cosmos), no. 34 (1896).

Black, M., and Y. Yacoob. “Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motions.” *ICCV*, 1995.

Bolle, R. C., H. H. Baker, and D. H. Marimont. “Epipolarplane image analysis: An approach to determining structure.” (IJCV) 1987.

Bouguila, L., M. Ishii, and M. Sato. “Effect of coupling haptics and stereopsis on depth perception in virtual environments.” *World Multiconference on Systemics, Cybernetics and Informatics* , 2000.

Bouguila, L., M. Ishii, and M. Sato. “What impact does the haptic-stereo integration have on depth perception in stereographic virtual environment? A preliminary study.” *Haptic HCI*, 2000.

Boykov, Y., O. Veksler, and R. Zabih. “Fast approximate energy minimization via graph cuts.” (IEEE TPAMI), no. 23 (2001).

Chaudhuri, S., and A. N. Rajagopalan. *Depth from defocus: a real aperture imaging approach.* Springer, 1999.

Chiu, K., M. Herf, P. Shirley, S. Swamy, C. Wang, and K. Zimmerman. “Spatially Nonuniform Scaling Functions for High Contrast Images.” *Procedings of Graphics Interface*, 93.

Cmglee. *Parallax barrier.* http://en.wikipedia.org/wiki/Parallax_barrier (accessed 05 30, 2012).

Cochran, S. D., and G. Medioni. “3-D Surface Description from Binocular Stereo.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14, no. 10 (October 1992).

Coles, T., and N. John. “The role of haptics in Medical Training Simulators: A survey of the state of the art.” *IEEE Transaction on Haptics* 4, no. 1 (2011).

Computer Arts 3D world glossary. 2011. ftp://ftp.futurenet.co.uk/pub/arts/Glossary.pdf (accessed 2011 03-11).

Cook, R., L. Carpenter, and E. Catmull. *The Reyes Image Rendering Architecture.* 1987.

Cook, R., T. Porter, and L. Carpenter. “Distributed Ray Tracing.” *Proceedings of SIGGRAPH*, 1984.

Cowan, M. *REAL D 3D Theatrical System. A Technical Overview.* Real D, 2007.

Del45. *Stereoscopy.* 08 10, 2009. http://en.wikipedia.org/wiki/Stereoscopy (accessed 05 30, 2012).

Dodgson, N. “Autostereoscopic 3D Displays.” (IEEE Computer), no. 38 (2005).

Durand, F., and J. Dorsey. “Fast bilateral filtering for the display of high dynamic range images.” *ACM Transactions on Graphics*, 2002.

Durand, F., and J. Dorsey. "Fast Bilateral Filtering for the Display of High Dynamic Range Images." *ACM Transactions of SIGGRAPH* (ACM), 2002.

Erker, G. *How To Freeview Stereo (3D) Images.* 2011. http://www.angelfire.com/ca/erker/freeview.html (accessed 2011 19-11).

Ernst, M. O., M. S. Banks, and H. H. Bulthoff. “Touch can change visual slant perception.” *Nature Neuroscience*, 2000 January: 69-73.

Feygin, D., M. Keehner, and F. Tendick. “Haptic Guidance: Experimental Evaluation of a Haptic Training Method for a Perceptual Motor Skill.” *Proceedings of the 10th Symposium on Haptic Interferences for Virtual Environments and Teleoperator Systems* 10 (2002).

Freedman, B., A. Shpunt, M. Machline, and Y. Arieli. Depth mapping using projected patterns. USA Patent 2010/0118123 A1. 2010 13-05.

Frisoli, A., G. Jansson, M. Bergamasco, and C. Loscos. “Evaluation of the pure-form haptic displays used for exploration of works of art at museums.” 2004.

Gernshein, H., and A Gernshein. *The History of Photography from the Camera Obscura to the Beginning of Modern Era.* New York: McGraw-Hill, 1969.

Glassner, A., K. Fishkin, D. Marimont, and M. Stone. *Device directed rendering.* Technical Report, Xerox, 1993.

Goedegebure, Sacha. *Big Buck Bunny.* Directed by Sacha Goedegebure. 2008.

Hannah, M. J. “Computer Matching of Areas in Stereo Images.” (Stanford University) 1974.

Hayward, V., and D. Yi. “Change of Height: An Approach to the Haptic Display of Shape and Texture Without Surface Normal.” In *Experimental Robotics VIII, Springer Tract in Advanced Robotics*, by B. Siciliano and P Dario, 570-579. 2003.

HIRO III. *Opposed-type Multi-Fingered Haptic Interface Robot III.* 2008. http://www.maru-tomi.co.jp/english/HIROE.pdf (accessed 2011 31-10).

Hong, L., and G. Chen. “Segment-based stereo matching using graph cuts.” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

Hong, L., and G. Chen. "Segment-based Stereo Matching Using Graph Cuts." *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE), 07 2004.

Ikits, M., and D. Brederson. “The visual haptic workbench.” In *Visualization Handbook*, by C. Hensen and C. Johnson, 431-447. Academic Press, 2004.

Isono, H., and M. Yasuda. Three-dimensional image display using electrically generated parallax barrier stripes. USA Patent 5315377. 1994 24-May.

Jebara, T.S., and A. Pentland. “Parametrized Stucture from Motion for 3D Adaptive Feedback Tracking of Faces.” *CVPR*, 1997.

Jorke, H., and M. Fritz. “Stereo projection using interference filters.” *Stereoscopic Displays and Virtual Reality Systems XIII*, January 2006.

Kawai, M., and T. Yoshikawa. “Stable haptic display of 1-dof grasping with coupling impedance for internal and external forces.” *Proceedings of the 2000 IEEE/RSJ*, 2000.

Kinect for Windows. 2011. (accessed 2011 20-11).

La Cascia, M., J. Isidoro, and S. Sclaroff. “Head tracking via robust registration in texture map images.” *CVPR*, 1998.

Laroussi, B., Y. Cai, and M. Sato. “New haptic device for human scale virtual environment: Scalable SPIDAR.” *ICMR*, 97.

Leptit, L, P Lagger, and P Fua. "Randomized trees for real-time keypoint recognition." *Proceedings of CVPR*, 2005.

Levin, A., R. Fergus, F. Durand, and W. Freeman. “Image and Depth from a Conventional Camera with a Coded Aperture.” *ACM Transactions on Graphics* 26 (July 2007).

Lippmann, G. “Épreuves réversibles. Photographies intégrales.” (Comptes Rendus de l'Académie des Sciences), no. 146 (1908).

Loscos, C., et al. “The Museum of Pure Form: touching real statues in an immersive virtual museum,.” *The 5th International Symposium on Virtual Reality, Archaeology and Cultural Heritage*, 2004.

Marr, D., and T. Poggio. “Cooperative Computation of Stereo Disparity.” *Science*, 1976 15-October: 283-287.

Microsoft Corporation. *Xbox 360 - Official Site - Xbox.com.* 2011. www.xbox.com/xbox360 (accessed 2011 21-12).

—. *Xbox 360 + Kinect.* 2011. http://www.xbox.com/en-US/kinect (accessed 11 20-11).

Morgenbesser, H. B., and M. A. Srinivasan. “Force shading for haptic shape perception.” *Proceedings of ASME Dynamic Systems and Control Division*, 1996.

Nechvatal, Joseph. *Immersive Ideals / Critical Distances.* LAP Lambert Academic Publishing, 2009.

Newman, R., Y. Matsumoto, S. Rougeaux, and A. Zelinsky. “Real-time stereo tracking for head pose and gaze estimation.” *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, 2000.

Nonvint. http://www.novint.com/index.php/products (accessed 2011 31-10).

Novint Technologies. *Falcon Technical Specifications.* 2011. http://www.novint.com/index.php/novintxio/41 (accessed 2011 20-11).

Novint Technologies Incorporated. "Haptic Device Abstraction Layer (HDAL) ." Programmer's guide, Albuquerque, NM, 2008.

Novint Technologies. *Novint Falcon.* 2011. http://www.novint.com/index.php/novintfalcon (accessed 2011 20-11).

Nvidia 3D Vision. http://www.nvidia.com/object/3d-vision-main.html (accessed 2011 25-10).

Nvidia. *NVIDIA 3D VISION.* 2011. http://www.nvidia.com/object/3d-vision-main.html (accessed 2011 19-11).

Okutomi, M., and T Kanade. "A multiple-baseline stereo." (IEEE TPAMI) 1993.

OpenCV. *Open CV.* 2011. http://opencv.willowgarage.com/wiki/.

Prime Sense. *Prime Sense - Natural Interaction.* 2011. http://www.primesense.com/ (accessed 2011 20-11).

Reach in. *Touch enabled solutions.* http://www.reachin.se/.

Roberts, J., O. Slattery, and D. Kardos. "Rotating-wheel Braille display for continuous refreshable Braille." *Digest of Technical Papers, Society for Information Display International Symposium, XXXXI*, 2000.

Robles-De-La-Torre, G. "Virtual Reality: Touch / Haptics." In *Encyclopedia of Perception*, by E. Goldstein. Sage Publications, 2009.

Rollmann, Wilhelm. "Zwei neue stereoskopische Methoden." (Annalen der Physik) 166 (1853).

Schödl, A., A. Haro, and I. A. Essa. *Head Tracking Using a Textured Polygonal Model.* Technical Report, Atlanta, Georgia: Georgia Institute of Technology, Graphics, Visualization and Usability Center, 1998.

Scharstein, D, and R Szeliski. "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondance Algorithms." (International Journal of Computer Vision) 2002.

Scharstein, D., and R. Szeliski. "High-Accuracy Stereo Depth Maps Using Structured Light." *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

Segal, M., and K. Akeley. *The OpenGL Graphics System: A Specification.* Silicon Graphics Inc, 1999.

Sensable. *Haptic Devices - Sensable.* 2011. http://www.sensable.com/products-haptic-devices.htm (accessed 2011 20-11).

—. *Haptic Devices & SDKs for Third-Party Development.* http://www.sensable.com/industries-application-development.htm (accessed 2011 30-10).

Shotton, J., et al. *Real-Time Human Pose Recognition in Parts from Single Depth Images.* Microsoft Research Cambridge & Xbox Incubation, 2011.

Sorensen, B., S. Hansen, and L. Sorensen. Method for recording and viewing stereoscopic images in color using multichrome filters. USA Patent 6687003. 2001 31-05.

Sutherland, I. *A head mounted three dimensional display.* Technical Report, Harvard University, 1968.

Swapp, D., V. Pawar, and C. Loscos. “Interaction with co-located haptic feedback in virtual reality.” 2005.

Szeliski, Richard. *Computer Vision: Algorithms and Applications (Texts in Computer Science).* Springer, 2010.

Thompson, T. V., D. D. Nelson, D. Nelson, E. Cohen, and J. Hollerbach. “Maneuverable Nurbs Models Within A Haptic Virtual Environment.” *6th Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 1997.

Toshiba . *News & Event : News Releases | Toshiba Mobile Display.* 2010 27-04. http://www.tmdisplay.com/english/news/2010/2010_0427.htm (accessed 11 20-12).

TriOviz. *INFICOLOR 3D Glasses.* 2010. http://us.trioviz.com/en (accessed 2011 20-12).

Tumblin, J., and H. Rushmeier. *Tone reproduction for realistic computer generated images.* Technical, Graphics, Visualization and Usability Center, Georgia Institute of Technology, 1991.

Vertrees, Jason. *Anaglyph 3D.* 11 15, 2011. http://en.wikipedia.org/wiki/Anaglyph_3D (accessed 05 30, 2012).

Wall, S. A., K. Paynter, A. M. Shillito, M. Wright, and S. Scali. “The effect of haptic feedback and stereo graphics in a 3D target acquisition.” *Proceedings of Eurohaptics*, 2002.

Ware, C., K. Arthur, and K. S. Booth. “Fish tank virtual reality.” *Proceedings of ACM INTERCHI*, 93.

Wheatstone, C. *Contributions to the Physiology of Vision.—Part the First. On some remarkable, and hitherto unobserved, Phenomena of Binocular Vision.* London: Experimental Philosophy in King's College, 1838.

Wihelm, R. “Zwei neue stereoskopische Methoden.” *Annalen der Physik 166* (Annalen der Physik), 1853: 186-187.

Williams, T. D. “Depth from camera motion in a real world scene.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-2 (November 1980): 511-516.

Yang, J., R. Stiefelhagen, U. Meier, and A. Waibel. “Real-time face and facial feature tracking and applications.” *Proceedings of AVSP*, 1998.

Yang, R., and Z. Zhang. “Model-based Head Pose Tracking With Stereovision.” *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, 2002.

Yobas, L., D. M. Durand, G.G. Skebe, F.J. Lisy, and M. A. Huff. “A novel integrable microvalve for refreshable Braille display system.” *Journal of Microelectromechanical Systems*, 2003.

Yoshida, T., et al. “RePro3D: full-parallax 3D display with haptic feedback using retro-reflective projection technology .” *IEEE International Symposium on VR Innovation*, 2011 .

Yoshikawa, T., Y. Yokokohji, T. Matsumoto, and X. Zheng. “Display of operating feel for the manipulation of dynamic virtual objects with frictional surface.” *Trans. ASME, Journal of Dynamic Systems, Measurement and Control*, 1995.

Zabih, R., and J Woodfill. “ Non-parametric local transforms.” (ECCV) II (1994).

Zilles, C., and J. K. Salisbury. “A constraint-based god object method for haptic display.” *Proceedings of IEEE/RSJ International Conference Intelligent Robots and Systems, Human Robot Interaction and Cooperative Robots*, 1995.