2013

Portal-s: High-resolution real-time 3D video telepresence

Nikolaus Lee Karpinsky
Iowa State University

Portal-s: High-resolution real-time 3D video telepresence

by

Nikolaus Lee Karpinsky

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Human Computer Interaction; Computer Engineering

Iowa State University

Ames, Iowa

2013

DEDICATION

I would like to dedicate this dissertation to my parents Kevin and Sharon, who instilled all of their knowledge onto me, fostering all of my gifts, making this dissertation possible. Mom and Dad, I love you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this dissertation. First and foremost, thanks to Dr. Song Zhang for his guidance and support throughout this research and writing. His guidance, enthusiasm, and patience has inspired me, continually renewing my interest and holding me to the highest set of standards. Thanks to Dr. James Oliver for all of his help, kindling my passion for my research, having great vision, and teaching me to share that vision with others. I would also like to thank my committee members for their efforts, guiding and encouraging me to completion: Dr. Eliot Winer, Dr. Jon Kelly, and Dr. Doug Jacobson. Lastly, a thank you goes out to my friends, my family, and my "Moose", who put up with me during my research and writing, forcing me to relax and grab a burrito when needed.

ABSTRACT

The goal of telepresence is to allow a person to feel as if they are present in a location other than their true location; a common application of telepresence is video conferencing in which live video of a user is transmitted to a remote location for viewing. In conventional two-dimensional (2D) video conferencing, loss of correct eye gaze commonly occurs, due to a disparity between the capture and display optical axes. Newer systems are being developed which allow for three-dimensional (3D) video conferencing, circumventing issues with this disparity, but new challenges are arising in the capture, delivery, and redisplay of 3D contents across existing infrastructure. To address these challenges, a novel system is proposed which allows for 3D video conferencing across existing networks while delivering full resolution 3D video and establishing correct eye gaze. During the development of Portal-s, many innovations to the field of 3D scanning and its applications were made; specifically, this dissertation research has achieved the following innovations: a technique to realize 3D video processing entirely on a graphics processing unit (GPU), methods to compress 3D videos on a GPU, and combination of the aforementioned innovations with a special holographic display hardware system to enable the novel 3D telepresence system entitled Portal-s.

The first challenge this dissertation addresses is the cost of real-time 3D scanning technology, both from a monetary and computing power perspective. New advancements in 3D scanning and computation technology are continuing to increase, simplifying the acquisition and display of 3D data. These advancements are allowing users new methods of interaction and analysis of the 3D world around them. Although the acquisition of static 3D geometry is becoming easy, the same cannot be said of dynamic geometry,

since all aspects of the 3D processing pipeline, capture, processing, and display, must be realized in real-time simultaneously. Conventional approaches to solve these problems utilize workstation computers with powerful central processing units (CPUs) and GPUs to accomplish the large amounts of processing power required for a single 3D frame. A challenge arises when trying to realize real-time 3D scanning on commodity hardware such as a laptop computer.

To address the cost of a real-time 3D scanning system, an entirely parallel 3D data processing pipeline that makes use of a multi-frequency phase-shifting technique is presented. This novel processing pipeline can achieve simultaneous 3D data capturing, processing, and display at 30 frames per second (fps) on a laptop computer. By implementing the pipeline within the OpenGL Shading Language (GLSL), nearly any modern computer with a dedicated graphics device can run the pipeline. Making use of multiple threads sharing GPU resources and direct memory access transfers, high frame rates on low compute power devices can be achieved. Although these advancements allow for low compute power devices such as a laptop to achieve real-time 3D scanning, this technique is not without challenges. The main challenge being selecting frequencies that allow for high quality phase, yet do not include phase jumps in equivalent frequencies. To address this issue, a new modified multi-frequency phase shifting technique was developed that allows phase jumps to be introduced in equivalent frequencies yet unwrapped in parallel, increasing phase quality and reducing reconstruction error. Utilizing these techniques, a real-time 3D scanner was developed that captures 3D geometry at 30 fps with a root-mean-square error (RMSE) of 0.00081 mm for a measurement area of 100 mm $\times$ 75 mm at a resolution of 800 $\times$ 600 on a laptop computer. With the above mentioned pipeline the CPU is nearly idle, freeing it to perform additional tasks such as image processing and analysis.

The second challenge this dissertation addresses is associated with delivering huge amounts of 3D video data in real-time across existing network infrastructure. As the speed of 3D scanning continues to increase, and real-time scanning is achieved on low compute power devices, a way of compressing the massive amounts of 3D data being generated is needed. At a scan resolution of 800×600, streaming a 3D point cloud at 30 frames per second (FPS) would require a throughput of over 1.3 Gbps. This amount of throughput is large for a PCIe bus, and too much for most commodity network cards. Conventional approaches involve serializing the data into a compressible state such as a polygon file format (PLY) or Wavefront object (OBJ) file. While this technique works well for structured 3D geometry, such as that created with computer aided drafting (CAD) or 3D modeling software, this does not hold true for 3D scanned data as it is inherently unstructured. A challenge arises when trying to compress this unstructured 3D information in such a way that it can be easily utilized with existing infrastructure.

To address the need for real-time 3D video compression, new techniques entitled Holoimage and Holovideo are presented, which have the ability to compress, respectively, 3D geometry and 3D video into 2D counterparts and apply both lossless and lossy encoding. Similar to the aforementioned 3D scanning pipeline, these techniques make use of a completely parallel pipeline for encoding and decoding; this affords high speed processing on the GPU, as well as compression before streaming the data over the PCIe bus. Once in the compressed 2D state, the information can be streamed and saved until the 3D information is needed, at which point 3D geometry can be reconstructed while maintaining a low amount of reconstruction error. Further enhancements of the technique have allowed additional information, such as texture information, to be encoded by reducing the bit rate of the data through image dithering. This allows both the 3D video and associated 2D texture information to be interlaced and compressed into 2D video, synchronizing the streams automatically.

The third challenge this dissertation addresses is achieving correct eye gaze in video conferencing. In 2D video conferencing, loss of correct eye gaze commonly occurs, due to a disparity between the capture and display optical axes. Conventional approaches

to mitigate this issue involve either reducing the angle of disparity between the axes by increasing the distance of the user to the system, or merging the axes through the use of beam splitters. Newer approaches to this issue make use of 3D capture and display technology, as the angle of disparity can be corrected through transforms of the 3D data. Challenges arise when trying to create such novel systems, as all aspects of the pipeline, capture, transmission, and redisplay must be simultaneously achieved in real-time with the massive amounts of 3D data.

Finally, the Portal-s system is presented, which is an integration of all the aforementioned technologies into a holistic software and hardware system that enables real-time 3D video conferencing with correct mutual eye gaze. To overcome the loss of eye contact in conventional video conferencing, Portal-s makes use of dual structured-light scanners that capture through the same optical axis as the display. The real-time 3D video frames generated on the GPU are then compressed using the Holovideo technique. This allows the 3D video to be streamed across a conventional network or the Internet, and redisplayed at a remote node for another user on the Holographic display glass. Utilizing two connected Portal-s nodes, users of the systems can engage in 3D video conferencing with natural eye gaze established. In conclusion, this dissertation research substantially advances the field of real-time 3D scanning and its applications. Contributions of this research span into both academic and industrial practices, where the use of this information has allowed users new methods of interaction and analysis of the 3D world around them.

CHAPTER 1. General Introduction

1.1 Overview

Advancements in three dimensional (3D) scanning technology and computation technology continue to increase, making the acquisition and display of 3D data simple (Zhang, 2010). These advancements in research have gone far enough to enter consumer markets, with the release of consumer 3D scanners such as the Microsoft Kinect (Microsoft, 2010). As advancements in 3D scanners continue to evolve, new research areas are emerging, such as research into real-time 3D scanners, 3D video compression, and utilizing the research in novel applications.

Although the acquisition of 3D data is simple for static geometry, the same is not true for dynamic geometry. In order to capture moving 3D geometry, 3D scanners must be able to sample an object at a high enough speed to ensure that the signal is not aliased. Challenges arise when trying to scale the speed of 3D scanners fast enough to prevent aliasing, as all aspects of the pipeline, capture, processing, and redisplay, must be able to keep up with the capture rate. One method to accomplish this is know as single-shot scanning, in which all needed data is captured in a single pass, and then 3D data is calculated from the snapshot (Beeler et al., 2010). Although beneficial for moving objects, it can be difficult to capture highly detailed 3D models in a single pass. Through the use of temporal multiplexing, multiple captures can be used to reconstruct a single 3D frame (Salvi et al., 2010). Using this method, highly detailed 3D models can be generated, but it requires high speed timing of the capture hardware to synchronize the various components. Recent advancements have utilized binary defocusing to reduce the needed timing precision, along with a graphics processing unit (GPU) to achieve processing and redisplay speeds (Li et al., 2010).

As the speed of 3D scanning enters real time and beyond, massive amounts of 3D data are generated, and a need for compression arises. Conventional approaches attempt to compress 3D geometry by storing all the vertices in a model, and then animating the model based on a subset of points that the vertices are constrained to (Forstmann et al., 2007). While this approach works well for structured 3D data, such as models designed in computer aided drafting (CAD) or 3D modeling software, this does not hold true for 3D objects captured from 3D scanners. Models and scenes captured from 3D scanners are inherently unstructured, since they are a sampling performed by the 3D scanner. In order to accommodate real-time 3D scanning for applications requiring the data, a new 3D compression technique named Holovideo has been developed that can compress and decompress 3D scanner data in real-time (Karpinsky and Zhang, 2012b). Through the use of virtual structured light scanning and parallel computation, Holovideo can compress 3D video in real-time with large compression ratios when compared to conventional 3D compression formats. This allows applications that need high fidelity 3D data yet efficient data transfer to utilize real-time 3D scanners.

One such application that requires real-time 3D scanners and real-time 3D video compression is the area of 3D telepresence. The goal of telepresence is to allow a person to feel as if they are present in a location other than their true location (Minsky, 1980). To achieve such a goal, technologies that can capture and transmit senses such as vision and hearing are utilized. Currently, common telepresence applications capture and transmit 2D video. With the advent of 3D scanning technology comes the idea of capturing and transmitting 3D video. To achieve this, research into real-time 3D scanning technologies and 3D video compression methods must be further advanced, as well as applying the research towards telepresence. To address this need of continued research, this dissertation presents our novel research on real-time 3D scanning, 3D video compression, and their application towards 3D telepresence.

1.2 Contribution of Research

This dissertation research has made breakthroughs in the field of 3D scanning, 3D information processing, and 3D telepresence. Specifically, this research has made the following major contributions:

1. **A framework to realize real-time 3D scanning entirely on a graphics processing unit (GPU).** In order to realize real-time 3D scanning, capture, processing, and redisplay must all be simultaneously achieved in real time. Previous works have utilized workstations with powerful central processing units (CPUs) to overcome the massive amounts of processing involved with such a task. More recently, other works have utilized a GPU in addition to the CPU to accomplish parts of the pipeline that can be realized in parallel (Zhang et al., 2006). To date, no other work has been able to utilize a GPU entirely due to serial computation in several steps of the pipeline. To overcome these limitations, an entirely new 3D processing pipeline has been developed which allows the GPU to perform all of the work in parallel and achieve real-time 3D scanning. This allows commodity hardware with modern GPUs to realize real-time 3D scanning. Specifically, a real-time 3D scanner was developed with a measurement speed of 30 frames per second (FPS) at a resolution of 800×600 pixels per frame on a laptop computer. This work has been published in the SPIE conference on Optical Engineering and Applications (Karpinsky et al., 2013a), and an extended version of this work has been submitted to the journal of Optical Engineering.

2. **A novel method to substantially compress 3D geometry data.** This technique creates a virtual structured light scanner to convert 3D geometries into 2D

images, and further store them in a lossy format. Three dimensional data coming from a structured light scanner is inherently unstructured due to the scanner performing a point sampling. Due to this unstructured nature, traditional 3D file formats that rely on structured 3D data are not well suited to compressing such scans. The Holoimage 3D file compression format was developed to overcome this problem. By utilizing a virtual structured light scanner, 3D data can be compressed into 2D images, and then later uncompressed back into 3D. This allows traditional image compression such as portable network graphics (PNG) compression to be utilized on 3D data, resulting in large compression ratios. Since the virtual scan uses sinusoidal fringe patterns, lossy image compression, such as joint photographic experts group (JPEG) compression, that utilizes the discrete cosine transform, can also be leveraged with little loss of information. This work has been published in the journal of Optical Engineering (Karpinsky and Zhang, 2010a).

3. **A technique to compress 3D video entirely on a GPU.** In order for data coming from a real-time 3D scanner to be saved and utilized later, the ability to compress the 3D video in real time must be realized. Achieving 3D video compression on the GPU is advantageous, as there is less information to transfer back to the CPU for streaming and saving. To accomplish this, the Holoimage technique was modified and extended into Holovideo which utilizes virtual structured light scanning, through the use of OpenGL shaders (GLSL). Using this technique, 3D video can be compressed by rendering to an offscreen render buffer, and then the resulting image can be either streamed across a network or saved to a file. This technique has been successfully used to save 3D video to different file formats such as QuickTime video, H.264 encoded video, and streamed across a standard network. This allows the data to be utilized with existing compression and network infrastructure, while still conveying the high resolution real-time 3D information. This work has been published in the journal of Optics and Lasers in Engineer-

ing (Karpinsky and Zhang, 2012b,a, 2013).

4. **A method to interlace additional information with compressed 3D video.** Real-time structured light scanners capture not only 3D information, but also other information such as texture information. Conventional approaches either discard this information, or serialize it into a separate stream and then reunifying the streams later during reconstruction; the former is undesirable as it involves loss of information, and the later as it can cause synchronization issues, resulting in visual artifacts. This problem is further compounded when using multiple reconstruction devices, such as systems with multiple GPUs or clusters of GPUs. To overcome this challenge, the Holovideo technique was extended utilizing image dithering, a technique common in digital printing for reducing the bit rate of images (Stucki, 1981). By dithering multiple sources of data, Holovideo frame and corresponding texture information, the bit rate can be reduced and therefore the resulting images can be interlaced in the bits of a conventional image. Applying this approach real-time 3D scanning, video frames and their associated texture files can be compressed in real time, and then streamed and saved. This work has been published in the journal of Applied Optics (Karpinsky et al., 2013b).

5. **Portal-s, a novel 3D video telepresence system that achieves correct eye gaze.** Telepresence is a long studied field that attempts to capture users senses, transmit, and reconstruct them at a remote end. Conventional approaches to videoconferencing, a form of telepresence, involve capturing vision using 2D video and reconstructing using a 2D display. While this allows a user to see from a specific vantage point, the user loses eye-contact when talking with another participant, due to the capture and display using two different optical axes. To alleviate this issue, the Portal-s system captures using dual real-time structured light scanners that share the same optical axis as the display. The separate 3D scans of the user

are then decoded on the GPU and merged to generate a single view of the user; geometry and associated texture information is then compressed with Holovideo compression and streamed across a network to a remote Portal-s node. At the remote node, the 3D geometry is reconstructed on a pane of Holographic glass for another user. This allows users of the system to achieve correct mutual eye gaze. This work is currently being submitted to the top computer graphics conference, the ACM SIGGRAPH conference.

1.3 Dissertation Organization

The remainder of this dissertation is broken up into nine chapters. Chapter 2 gives a comprehensive literature review of 3D scanning techniques, 3D compression, and application of 3D scanning to telepresence. Chapter 3 reviews various fringe projection techniques, such as single step, dual step, and three-step fringe projection. Chapter 4 presents taking a multi-wavelength three step fringe projection technique and realizing it in real-time on current commodity hardware. Utilizing this technique a real-time structured light scanner was developed on a laptop with a resolution of 800×600 at 30 fps.

Chapters 5-8 address real-time 3D geometry video compression. Chapter 5 introduces a technique for 3D shape compression using a virtual structured light scanner entitled Holoimage. Using this technique, 3D scans can be compressed with little error. Chapter 6 expands the previous technique into 3D video creating a technique entitled Holovideo. The system is able to compress and decompress 3D video in real-time, with a compression ratio of over 134:1 when compared to the Wavefront object (OBJ) file format. Chapter 7 generalizes the work of Chapter 6 to more advanced video codecs that make use of the YUV colorspace, namely the H.264 codec. Utilizing H.264 compression, a 352:1 lossless compression ratio when compared to the OBJ file format is achieved, and a 6086:1 lossy compression ratio is achieved. Chapter 8 extents the Holovideo technique to address another challenge, compressing additional scan information, namely a models texture. Compression ratios of over 8.2:1 when compared to standard Holovideo compression have been achieved with a mean squared error of 0.34%, while transmitting additional texture information.

Finally, Chapter 9 presents how all of this previous research has been integrated into a telepresence system. Through the use of dual real-time structured light scanners, real-time 3D data merging on the GPU, and Holovideo compression, the Portal-s telepresence system is created. It has the ability to capture a user in 3D through the same optical axis that it displays to, establishing correct mutual eye gaze. Lastly, Chapter 10 summarizes this dissertations research, and presents some future directions to further advance or leverage this dissertation research.

1.4 Conclusion

Advancements in 3D scanning and computational technology have accelerated to the point of making the acquisition and display of 3D data simple. These advancements have led to new research areas, such as real-time 3D capture, 3D video compression, and the utilization of such technologies. This dissertation research work on 3D scanning has enabled the entire 3D pipeline to be achieved in parallel, allowing it to be performed on parallel compute devices such as a GPU. This has allowed real-time 3D scanning to reach commodity computing devices such as laptops, filling the need for low cost yet accurate 3D scanners. This dissertation research innovation on 3D video compression has achieved real-time 3D video compressing, allowing high-fidelity 3D information to be streamed and saved. Lastly, telepresence has benefited from these research advancements. Through the development of Portal-s, users are able to achieve correct mutual eye gaze during video conferencing.

CHAPTER 2. Review of literature

The goal of telepresence is to allow a person to feel as if they are present in a location other than their true location; this is typically achieved through the use of technologies that can transmit senses such as hearing, vision, etc. (Minsky, 1980). Due to the complexity involved with achieving such a goal, telepresence itself is a large and interdisciplinary field, comprised of a variety of knowledge bases. Today, the most common application of telepresence is videoconferencing, via consumer applications such as Skype or FaceTime. These applications allow a user's image and voice to be captured with a two-dimensional (2D) video camera and microphone, have the data compressed and streamed, and then reconstructed at another location. By achieving real-time speeds, these applications allow their users to communicate in real time, both visually and verbally.

As described above, the actual telepresence process can be defined through four stages: capturing, encoding, decoding, and redisplaying. Traditionally, these stages are done in 2D, using a 2D camera for capture, encoding and decoding using 2D video codecs, and then redisplaying 2D information in a remote location. Though the results are impressive and valuable, they are not without problems. The major problem is the loss of correct eye gaze. As a result of the capture and display technology having different optical axes, loss of eye gaze occurs even if both parties are looking directly at one another within their respective digital representation. Furthermore, because the capture position is typically fixed, as a result of a stationary 2D camera, it is difficult to accommodate for different head positions during redisplay; and this results in singular view rendering. These limitations hinder the overall effectiveness of making the user feel as if they are present in a location other than their true location. This chapter will present relevant literature addressing these issues. Prior research in the disciplines relating to three dimensional (3D) scanning, 3D compression, and 3D telecommunication will be addressed. An introduction to each discipline will be given, along with discussion of relevant literature in terms of advantages and disadvantages for each technique.

2.1 3D Scanning

Acquiring 3D geometry is a long studied field with numerous techniques for acquiring 3D coordinates based on real world objects. 3D scanning techniques can be classified into two different categories, contact and non contact (Curless, 1999), shown with the taxonomy in Figure 2.1. Contact 3D methods measure and reconstruct 3D geometry by probing the 3D surface through physical touch. An example of such a technique is a coordinate measurement machine (CMM) that can measure 3D geometry through a precise carriage system or articulated probe arm (Hocken, 1995). While this type of measurement can achieve high accuracy, it is typically limited to acquiring a small number of points per second, since the system must physically probe and touch the object point by point. Furthermore, due to the necessity of physical contact, it is undesirable for soft or deformable object measurement, or in the case of this work, measuring a dynamically changing human face.

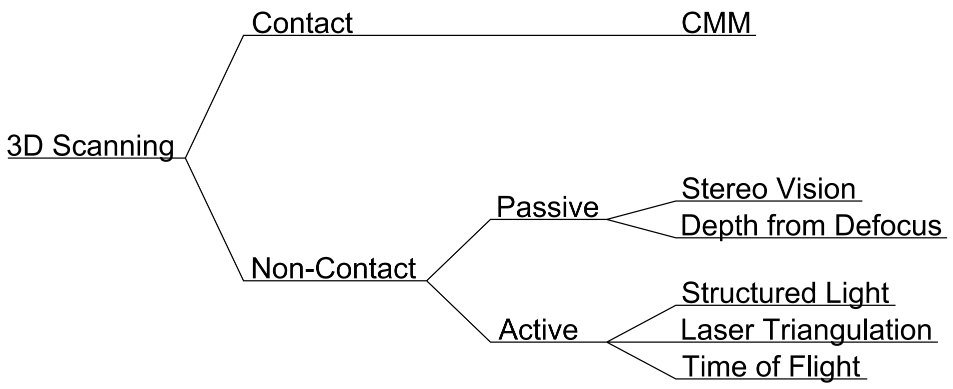


Figure 2.1 Taxonomy of 3D scanning systems

The non-contact 3D scanning technologies can be broken down further into passive and active methods (Curless, 1999). Passive methods do not emit any type of radiation. Rather, they rely on ambient radiation, passively observing an object and inferring its shape. Technologies in this class include stereo vision, depth from focus/defocus, and others. Unlike passive systems, active systems emit some type of radiation be it visible, infrared, or laser light and infer an object's shape by how the surface geometry distorts this radiation. Techniques in this category include time of flight, laser triangulation, and structured light.

2.1.1 Passive 3D Scanning

As previously stated, passive 3D scanners do not emit any type of radiation, but instead passively observe an object or scene under ambient radiation and infer its shape. Many optical based 3D scanning systems make the assumption that all depth ranges in a scene are in focus simultaneously. In actuality, this is a false assumption since in practical optical systems, only a limited depth range is in focus at a given time. Depth from focus/defocus systems are a form of passive 3D scanning technique that work off this principle (Subbarao and Surya, 1994). By measuring how defocused various regions of a scene are under varying focal lengths, depth from defocus systems can determine depth. Figure 2.2 illustrates an example of this principle. Moving from (a)-(c) the focal length is being pushed farther from the imaging plane. The first object to come into focus is the statue, thus it is the closest. Next comes the Rubiks cube, and then the end of the scene.

The limitations of this technique for resolution and speed depend on the resolution of the detector and accuracy and speed of the defocus measurement. Quantizing the level of defocus can be achieved in one of two ways: either making a direct measurement of defocus and then determining depth from the measurement (Aslantas, 2007), or by changing the intrinsic properties of the camera, namely focal length, and then measuring when regions are in focus (Favaro and Soatto, 2005). By making a direct measurement of defocus to determine depth, real-time speeds can be achieved, but this results in lower resolution as defocus is calculated by fitting a defocussing function compared to observing multiple levels of defocus. Changing intrinsic properties of camera such as focal length and then measuring focus results in much higher resolution, but slower speeds as multiple images have to be taken while changing physical camera characteristics in between each exposure. Finally, both of the these techniques depend on strong texture variation to measure focus/defocus, thus close to uniformly textured objects such as the statue in Figure 2.2 are difficult to measure.

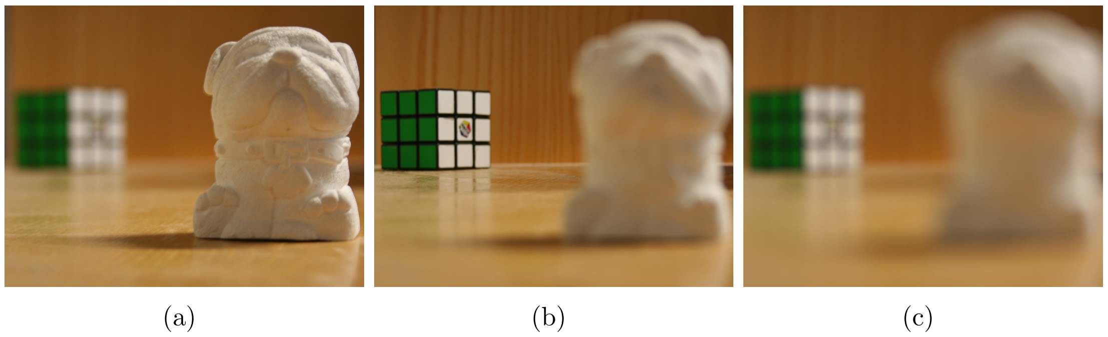


Figure 2.2 Principle behind depth from defocus. (a) the statue is in focus thus it is the closest in the scene, (b) the Rubiks cube is in focus thus it has a farther depth, (c) farthest focal length, no object is clear in the scene.

Another approach to passive 3D scanning, similar to how humans extract 3D information, is stereo vision. Stereo imaging based systems work by analyzing two or more views of the same scene from different perspectives (Scharstein and Szeliski, 2002). Assuming that the relationship of different perspectives can be determined, typically through a stereo matching, 3D geometry can be reconstructed through triangulation. Figure 2.3 demonstrates the concept of stereo triangulation. Some point, X_L, is found in the left camera, C_L. Through calibration, we know the position of the camera's principle point and can then form a line using X_L and the principle point. By imaging the same scene with another camera, C_R, we can then find the same point, X_R, and again form a line using the principle point. The intersection of the two lines then gives us the real world position of the 3D point. As the pixel location of the point changes in the right camera, C_R, different 3D points are triangulated: X_1, X_2, or X_3. The limiting factor on determining the accuracy is dependent on multiple factors, the camera's resolution, the accuracy of stereo calibration, and the ability to determine corresponding feature points.

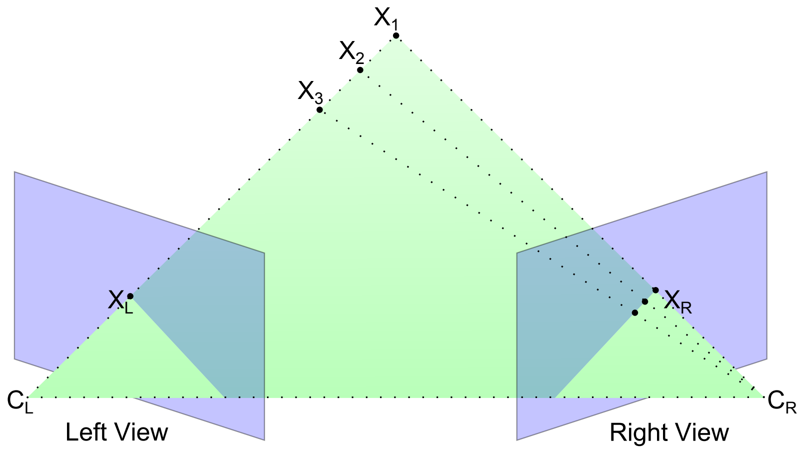


Figure 2.3 Conceptual diagram of stereo based triangulation. Cameras C_L and C_R image the same scene from different perspectives. Assuming corresponding points X_L and X_R can be found, resulting point X_1 can be triangulated. As X_R shifts farther down, different points X_2 and X_3 would be triangulated.

Stereo calibration is a widely studied topic, with various approaches, but is predominately performed by analyzing a checkerboard under various poses (Tsai, 1987; Zhang, 1999, 2000). As previously stated, this is performed prior to measurement, in order to determine intrinsic and extrinsic camera properties. Intrinsic properties are those relating to the camera, such as focal length, skew factor, and principle point, which are used to project from 3D lens coordinates to the 2D camera image plane; extrinsic properties are those relating to the cameras position in real world space and are used to transform between 3D world space and 3D lens space. Figure 2.4 shows an illustration of this concept. Once intrinsic and extrinsic camera properties are determined, the projection of a 3D point can be described as

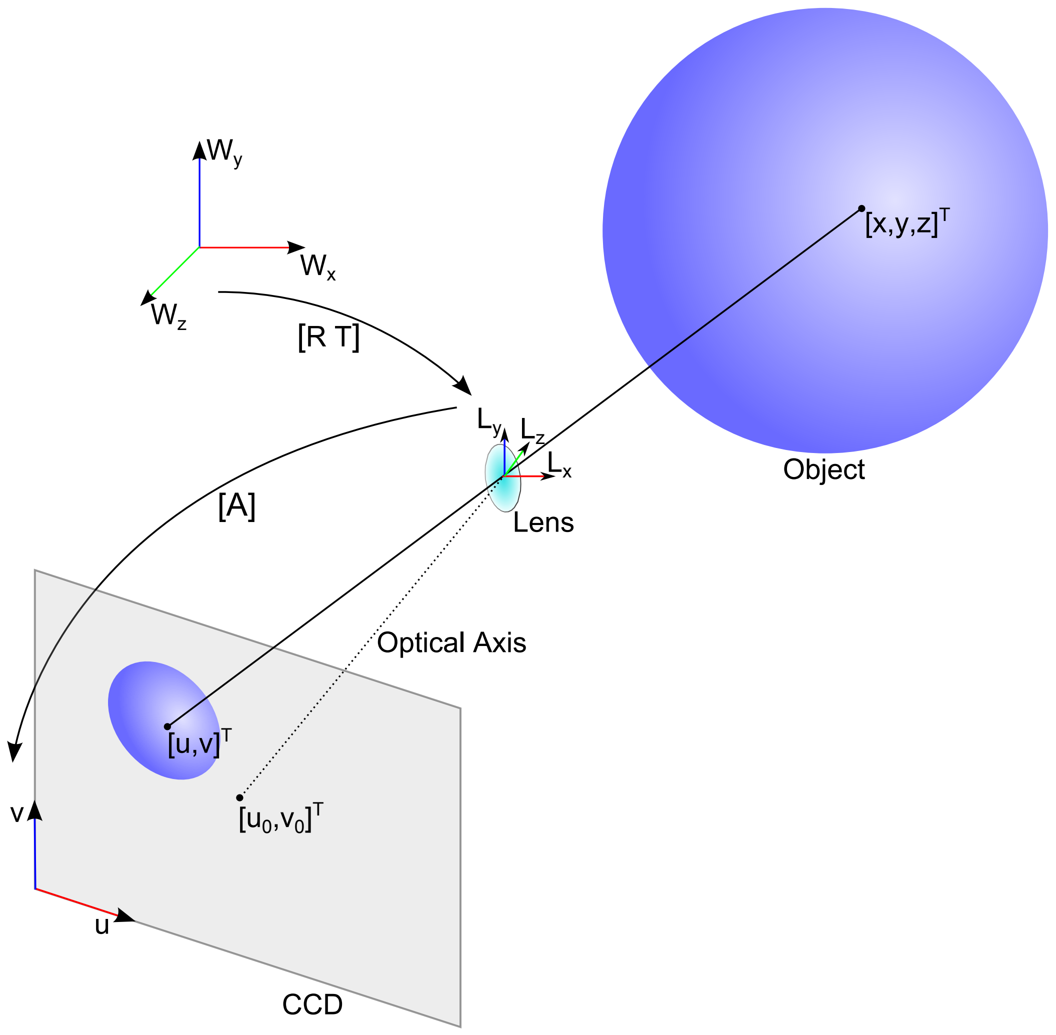


Figure 2.4 Illustration of pinhole camera calibration. Extrinsic properties pertaining to rotation and translation transform 3D world coordinates into 3D lens coordinates. Intrinsic properties pertaining to focal length, skew factor, and principle point project 3D lens coordinates onto the 2D camera image plane. The goal of camera calibration is estimate both intrinsic and extrinsic properties so that views from multiple cameras can be rectified together.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.1}$$

Here, (u_0, v_0) represent the cameras principle point, α and β represent the focal lengths in the u and v axis, γ represents the skew factor, and $[R\ t]$ represent the extrinsic parameters. If the correspondence is matched between two cameras, a set of four linear equations describing an over determined problem can be solved in the least squared sense to yield (X, Y, Z) information (Ogale and Aloimonos, 2005).

2.1.2 Active 3D Scanning

Unlike passive systems, active systems emit some type of radiation, that could be visible light, infrared light, or some other form of radiation, which is used to infer an object's shape (Besl, 1988). Technologies in this category include time of flight, laser based triangulation, and structured light. Time-of-flight-based systems measure depth by actively illuminating a scene with a light source and then recording the time it takes to observe the reflected light (Oggier et al., 2004). If an accurate measurement of the time it takes for light traveling from the emitter to the detector is known, the distance can be measured. Figure 2.5 illustrates the principle behind the technique. To accurately determine the time it takes to travel from the emitter to the detector, a light with a modulating phase is typically employed (Lange and Seitz, 2001). By modulating the phase at a constant frequency, and then recording the phase with the detector, the time can be accurately determined using the phase difference. The key component in a time-of-flight system is a detector that can measure intensity and distance information to the target in parallel without scanning. The limiting factor of the system is the resolution of the detector, and resolution of the phase measurement. Most time-of-flight systems are limited in sensor resolution, due to the requirement of measuring both intensity and distance information, resulting in low resolutions of approximately 200×200.

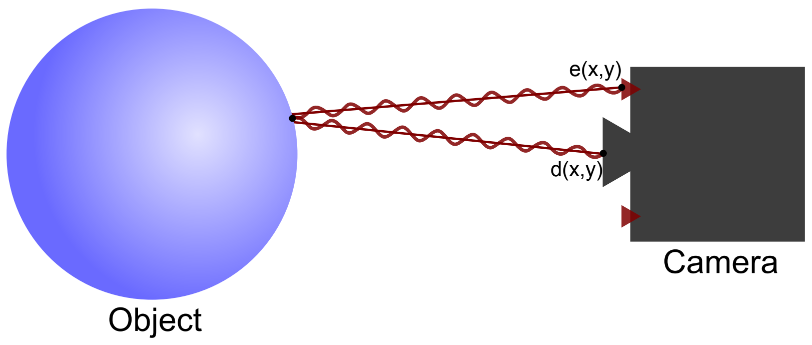


Figure 2.5 Conceptual diagram of a time-of-flight system. Emitter emits phase modulated signal from point $e_{(x,y)}$ which hits the object and then is reflected back to the detector $d_{(x,y)}$. Using the phase difference the time it took the signal to move from the emitter to the detector can be determined, which is then used to calculate the distance Z to the object in question.

Laser based triangulation systems work by actively illuminating a point or stripe on the object being scanned from one perspective, and then capturing from another angle. Figure 2.6 illustrates the principle behind the technique. The emitter or laser projects either a point or stripe of light onto the subject, and a camera or CCD array captures the resulting scene from another perspective. Similar to stereo vision, if the relationship between the emitter and detector can be determined, 3D points can be triangulated. Since two intersecting lines provide a 3D point, a laser stripe can be projected onto the object to simultaneously measure multiple points in the line, speeding up the process. With high quality calibration, sub micrometer accuracy can be achieved, making laser based systems a good candidate for applications needing highly accurate measurement such as scanning of historical artifacts (Bernardini et al., 2002; Remondino, 2011). The limiting factor in such systems is the speed at which the laser stripe can be swept across the object and detected. Commercial systems, such as the Leica ScanStation are limited to approximately 1 million points per second (Leica, 2012), thus difficult for applications requiring high resolution and high speed such as measuring detailed facial expressions.

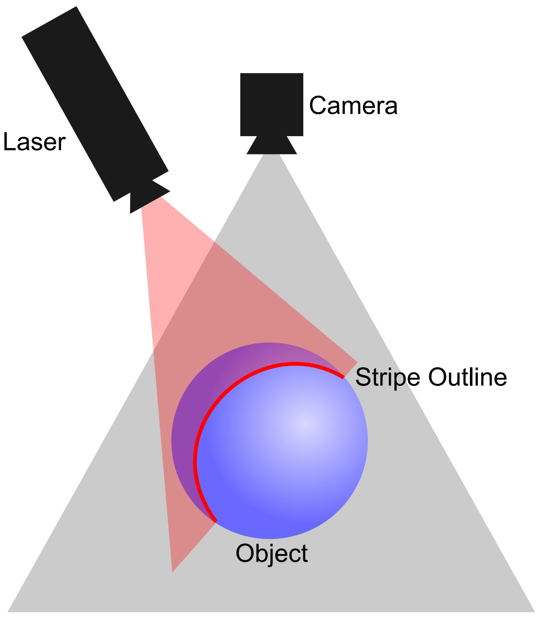


Figure 2.6 Conceptual diagram of a laser based triangulation systems. The laser emits a laser stripe onto the object and is the recorded by the camera from another angle. By knowing the relationship of the laser to the camera, 3D coordinates can be triangulated.

2.1.3 Structured Light Scanning

Structured light based scanning techniques are similar in nature to a stereo image based approach, except that in the fundamental system, one of the cameras is replaced with a structured light emitter. This emitter is used to project a form of coded structured light (CSL) onto the object that can then be used to establish the correspondence between the emitter and detector. This effectively gives a view of the scene from two perspectives with a correspondence between the views, circumventing the correspondence problem that is typically seen in passive stereo vision systems. Figure 2.7 illustrates the concept behind structured light scanning. In this diagram, the left camera is replaced by the emitter, in this case a projector, that projects the CSL onto the object being scanned. The detector, in this case a camera, on the right captures the resulting scene. From the perspective of the detector, the pattern appears geometrically distorted due to the shape of the object being scanned (Gorthi and Rastogi, 2010). Assuming the relationship between the camera and projector is known, 3D geometry can be recovered through triangulation (Zhang and Huang, 2006a). The limiting factor in such systems is the speed at which the pattern can be decoded along with the resolution of the emitter and detector. These factors can be addressed through pattern codification and calibration of the system. Current systems have been able to reach sub millimeter resolution at speeds of over 30 fps with resolutions of 640×480, making structured light scanning a suitable choice for high-resolution real-time applications such as 3D video conferencing (Zhang, 2010).

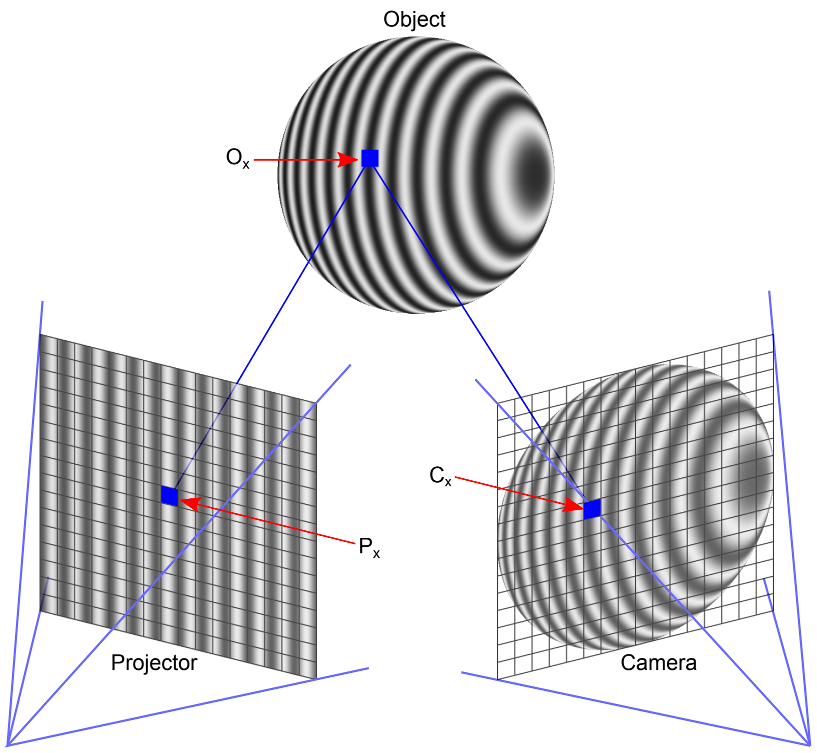


Figure 2.7 Conceptual diagram of structured light scanning. The emitter, a projector in this case, projects a form of CSL onto the object which is the captured by the detector, a camera in this case. If a correspondence can be achieved with a projector point P_x and a camera point C_x the depth at the resulting object point O_x can be decoded.

2.1.3.1 Pattern codification

Many different types of patterns for codification can be selected, each with their advantages and disadvantages. Salvi et al. (2004, 2010) have developed a taxonomy of different CSL techniques with the main distinction being between discrete and continuous patterns. Discrete patterns involve uniquely coding pixels in the projector through a discrete pattern, with techniques such as binary codes, gray codes, De Brujin codes, and M-array codes. The simplest of discrete codifications is binary codes in which pixel is composed of a unique codeword of 1's and 0's. To determine the code word for a pixel, images are sequentially projected and captured, then each image pixel is quantized into either 1 or 0, and the result is stored in a bit of the codeword. Thus, binary codes are limited to coding 2^n unique codewords for n images. While high speeds can be achieved using discrete CSL techniques such as binary codes, they are limited by both the resolution of the emitter and detector being used. Discrete signals need to be both projected and captured, limiting the overall resolution to being less than or equal to a single projector as well as a camera pixel.

Continuous CSL patterns on the other hand are continuous signals that cover the object being scanned. Since they are continuous signals, they are not limited by the projector's resolution; a discrete image can be projected by a slightly defocused projector resulting in a continuous signal covering the object. Thus the resolution is only limited by the camera used for reconstruction, being desirable for high resolution measurement. Different types of patterns exist such as single sinusoidal, single phase shifting, multiple phase shifting, frequency multiplexing, and spatial multiplexing (Karpinsky and Zhang, 2010b). Each of these codification strategies have advantages and disadvantages with regards to being able to measure specific surface types and requiring a number of patterns to be projected and captured to establish the correspondence. Phase-shift patterns involve projecting and capturing a series of phase shifted sinusoidal patterns. Through analysis of the recovered patterns, a process known as phase wrapping and unwrapping, the original phase value can be recovered providing a codification of pixels through phase. An example set of three patterns and recovered phase value is shown with Figure 2.8. Typically only a few patterns are needed for codification, thus phase-shifting techniques are beneficial for high-speed applications.

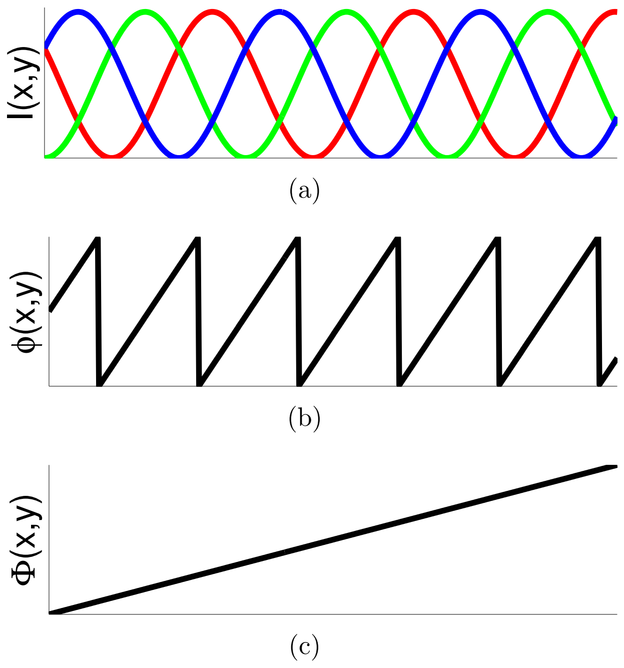


Figure 2.8 Example cross section of phase-shifting based CSL. (a) shows three phase shifted patterns encoded as red, green, and blue waveforms. (b) shows ϕ obtained from phase wrapping. (c) shows Φ obtained from phase unwrapping which is then used for codification of pixels.

2.2 3D Compression

Assuming that 3D information can be obtained in real time, the 3D information must be compressed before it can be streamed or utilized. Traditionally, computer graphics has employed many different techniques, such as Wavefront object (OBJ) files or polygon file format (PLY), for storing models and their associated data such as normals, UV coordinates, etc. Although these techniques work well for structured meshes with predefined animation or for static scans, the same does not hold true for high-resolution real-time

3D scans. Geometry coming from 3D scanners is inherently unstructured as it is a form of point sampling, thus requiring special forms of storage. To address this challenge, different approaches have been developed such as heuristic based encoding (Gumhold et al., 2005; Merry et al., 2006) and image based rendering approaches (Krishnamurthy et al., 2001; Gu et al., 2002).

Image based rendering approaches work well, as the 3D geometry from the scanner can be projected into 2D images, typically from the 3D scanners perspective, and then written to a compressed format. Once the geometry is projected, 2D image and video compression can be leveraged to compress the 3D data. Later when the 3D geometry is needed it can be recovered from the 2D images using image based rendering. This section will investigate different techniques for compressing raw 3D range data including: Octree Based Compression, Heuristics Based Compression, Depth mapping, and Geometry Images.

2.2.1 Octree Based Compression

Octree based compression involves representing 3D geometry or scan data as an Octree, and then serializing it out into a compressed state (Schnabel and Klein, 2006). This approach has the advantage of being able to find, add, remove, and modify data quickly $O(n \log n)$ where n is the number of points in the Octree (Elseberg et al., 2011). This advantage of traversal and modification has allowed Octree implementations to see exposure in computer graphics with ray tracing (Levoy, 1990), and 3D image processing in surface reconstruction and registration algorithms such as ICP (Besl and McKay, 1992; Rusinkiewicz and Levoy, 2001).

The Octree approach starts by creating a head node, which has eight children nodes. Each child node is a uniform subdivision of 3D space; an initial representation of this is shown with Figure 2.9 (a). Each child node or child subspace is then divided up into 8 uniform subspaces until either the subspace contains 0 - 1 points, or the traversal depth is reached. At this point the data is held by the child leaf node. Figure 2.9 (b) shows a subdivided Octree with a depth of 5.

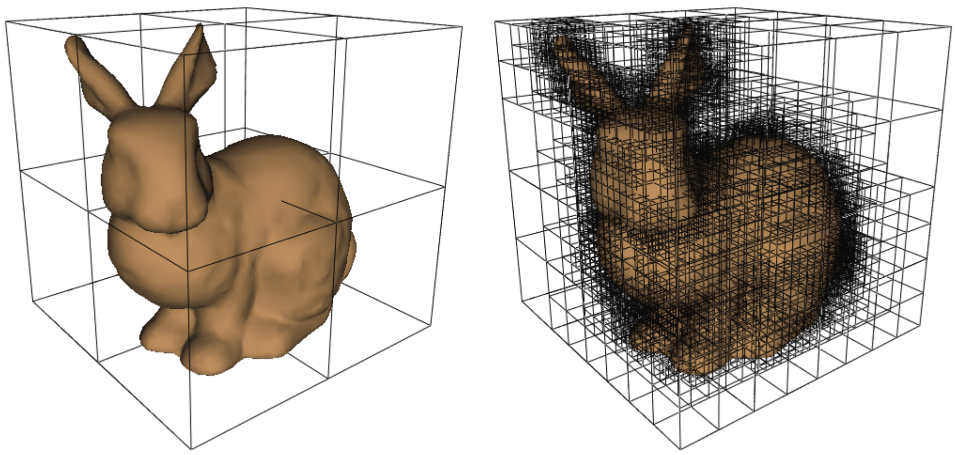

(a) (b)

Figure 2.9 Example of an Octree encoding performed on a scan of the Stanford bunny. (a) Octree with a max depth of 1. (b) Octree with a max depth of 5.

Since Octrees subdivide space, if it can be determined that a specific subspace does not need to be drawn, all the children in that subspace can be culled as well. This makes them very efficient with good scene graph and culling algorithms, that determine whether or not a specific subspace needs to be drawn. This advantage makes them extremely efficient in redisplay, allowing Octrees to see wide exposure in static 3D scanning. Although efficient for static 3D scanning, they become too cumbersome and slow for real-time capture data.

2.2.2 Heuristics Based Compression

Heuristic based point cloud compression is a lossless form of point cloud compression. The theory behind the technique is to start with a point in the point cloud and then move to a predicted point based off a set of heuristics, and encode a correction vector that moves the predicted point to the nearest point. To decode, the decoder starts again at the initial point, uses the heuristics to predict the next point, and then corrects using the supplied correction vector. Different techniques use different sets of heuristics, with the better methods being able to more precisely predict the next point requiring smaller correction vectors (Gumhold et al., 2005; Merry et al., 2006).

The main challenge in these techniques involve developing good heuristics that correctly predict the next point in the sequence. If bad predictions occur, large correction vectors are needed, resulting in little compression. Achieving high compression speeds is also difficult, since the algorithm is a serial algorithm, starting at one point and moving to another. Though the use of efficient data structures speeds of up to 500K points per second have been achieved (Gumhold et al., 2005). For real-time high-resolution 3D scanning this is not fast enough, as encoding will take longer than acquisition, resulting in a bottleneck, slowing down the frame rates.

2.2.3 Depth Mapping

To overcome the challenge of compressing and transmitting large amounts of 3D data, computer graphics has employed depth mapping for sometime. The idea behind depth mapping is to encode 3D geometry into 2D images, which can then later be decoded back to 3D with the use of image based rendering (Krishnamurthy et al., 2001). To encode a 3D scene into a 2D image, the technique divides a scene with a uniform grid of X and Y values, where Z values are encoded as the color value of the image. An analogy to this is using a camera to image a scene, with the color representing the depth Z and the pixel location holding the X and Y values. In this technique, the X and Y values are assumed to be uniform and are resampled if needed, only depth from a single perspective is encoded, and the depth resolution is dependent on the color resolution. For floating point image formats, the depth resolution is floating point, and for 8 bit images the depth resolution is limited to 256 levels.

Due to the nature of most 3D scanners generating 3D information from a single perspective, a natural connection with depth mapping is apparent. The technique is also desirable since it is easy to depth map a scene using the OpenGL Shader Language (GLSL) or by simply retrieving the OpenGL depth buffer and quantizing it into an image. Both approaches result in extremely fast and parallel encoding on modern graphics processing units (GPU). To decode a 2D image back into 3D information, the depth map can be sent along with an array of vertices to the GPU, where the vertex depth Z position can be adjusted again in GLSL vertex shading operations. Since both encoding and decoding can be realized in parallel, this technique is extremely fast, making it desirable for real-time encoding and decoding. These advantages have made it the encoding technique of choice for many 3D scanners such as the Microsoft Kinect (Microsoft, 2010).

2.2.4 Geometry Images

Developed by Gu et al. (2002), Geometry Images is another technique that was developed to compress 3D geometry into 2D images. The premise behind the technique is to make a series of cuts on a 3D model and then unwrap the model onto a flat disk. Each pixel on the disk corresponds to a single 3D point. Each of these points (x, y, z) then gets encoded as (r, g, b) within an image, effectively projecting 3D geometry into 2D images. Once in the 2D image state, the geometry can be compressed with traditional image compression algorithms, or video compression algorithms (Briceño et al., 2003).

To recover the 3D information, geometry images can be transferred to the GPU as a texture along with an array of vertices, which can then be decoded back to the 3D form using GLSL shaders. To achieve level-of-detail rendering, geometry images can make use of GPU functions such as mip-mapping to naturally downsample the geometry. Also, since decoding can be done entirely on the GPU, it can be leveraged in other parallel operations such as ray tracing (Carr et al., 2006). Although geometry images have many merits such as being able to encode 3D manifold geometry and achieving level-of-detail rendering through standard GPU functions, it is limited by its encoding size. As the genus of the model to be encoded increases, more cuts are needed which takes long periods of time to find. Even with low genus models, less than six, with a face count of less than 300K it takes approximately an hour to encode. This makes real-time encoding infeasible, thus they are not applicable for real-time 3D telepresence.

2.3 3D Telecommunication

The application of 3D telecommunication has been a long-standing goal in the field of telecommunication, but daunting challenges in both hardware and software limit implementation to few. In order to achieve 3D telepresence and address the goal of allowing a person to feel as if they are present in a location other than their true location, systems need to simultaneously achieve real-time 3D scanning, transmission, and redisplay. One of the most prominent works that addressed this challenge was *the office of the future* created at the University of North Carolina Chapel Hill (Raskar et al., 1998). The authors addressed and uncovered many of the technical challenges associated with real-time 3D telepresence and demonstrated various pieces of technology that could provide solutions to these challenges.

Following these visions, Gross et al. (2003) from ETH Zürich developed blue-c, a spatially immersive 3D video conferencing portal. By combining stereo reconstruction with unique CAVE^{TM}wall design, the authors were able to create a 3D video conferencing portal that could both display in stereo 3D and capture its user. Although this technique can solve problems with establishing eye contact and correct view dependent rendering, technical limitations hindered the overall experience, resulting in low quality resolution and speed. In 2009, Jones et al. (2009) developed a one-to-many 3D video conferencing system that achieved bi-directional eye contact. Using 3D, the system was able to achieve eye contact and view independent rendering for the audience, but a 2D limitation for the user being scanned resulted in singular view rendering for the user. The remaining section of this chapter will further investigate this research, giving more details on their implementation and results.

2.3.1 Office of the future

The office of the future is an attempt at creating a system that could be installed into an existing office to provide a seamless 3D telepresence system (Raskar et al., 1998). The system combines structured light scanning with spatially immersive displays created from a combination of CAVE$^{\text{TM}}$like walls and every-day surfaces. Through the use of capture and display systems that cover the entire work surface, users can have a traditional "untidy" desk with objects that can be simultaneously scanned and displayed on, through automatic calibration.

To achieve 3D capture, *the office of the future* makes use of a structured light system comprised of multiple cameras and projectors. The end goal of the system was to use the multiple projectors and cameras together with imperceptible binary structured light to achieve real-time scanning. Due to hardware limitations during the creation of the system, the authors were unable to reach real-time speeds. Even with the constraints, the system was able to demonstrate the use of imperceptible structured light at 3 Hz, which proved fast enough for semi-static object capture. This was then used to provide automatic calibration of the surfaces in the office so that they could be used as spatially immersive displays.

Display within the office was achieved by an array of five projectors projecting onto the wall displays, desk, and everyday objects within the office. By knowing the 3D geometry of the office through the structured light scan, display could be achieved through a two-pass rendering scheme. Initially, the desired image to be displayed on the office is rendered to a texture. Next the texture is projected from the workers view, which is tracked by a magnetic tracker, onto the geometry of the office from the scan, and the resulting scene is rendered out. This technique of automatic calibration of objects and displays within the office allows the worker to move objects yet still achieve 3D telepresence with another location. Although the authors were not able to achieve a fully complete and working system due to technical limitations at the time of publishing, they have laid out ideas which could be realized with future work.

2.3.2 Blue-c

The blue-c telepresence system (Gross et al., 2003) is an immersive 3D based telepresence system that attempts to solve bi-direction projection and acquisition. It combines stereo reconstruction along with a CAVE^{TM} (Cruz-Neira et al., 1993) based display system to create an immersive telepresence system for collaboration. Blue-c achieves bi-directional projection and acquisition through the use of active walls (Kunz and Spagno, 2004; Lampert, 1999) that can be electrically shuttered from opaque to transparent, effectively multiplexing acquisition and display in time. By making the active walls opaque during a projection cycle and transparent during a acquisition cycle, acquisition and display can be achieved through the same optical axis, realized as the screen normal.

To achieve 3D capture, blue-c makes use of stereo reconstruction that occurs during its acquisition cycle. During the acquisition cycle, the walls become transparent, the user's shutter glasses become opaque on both eyes, and an array of LEDs light the inside of the blue-c portal. An array of 16 cameras then capture the inside of the blue-c portal, scanning the user within. The video processing pipeline performs a silhouette-based approach to segmentation and 3D reconstruction of the user, similar to the visual hull stereo reconstruction proposed by Matusik et al. (2001). The pipeline is able to achieve speeds of up to 9 fps for 15K points or 5 fps for 25K points on the user, limiting the overall immersiveness of the reconstruction.

Blue-c makes use of a custom API for 3D video and sound built on top of a distributed scene graph to achieve 3D streaming. Each blue-c portal is then connected to a 100Mbps network and nodes update their display from the distributed scene graph. The scene graph is based on SGI OpenGL Performer (Rohlf and Helman, 1994) due to it natively supporting multi-processed rendering and high throughput. Finally, to achieve 3D redisplay, blue-c makes use of active stereo shutter glasses with a magnetic tracker to reconstruct correct view dependent rendering for the user in the CAVE^{TM}. Redisplay occurs while the active walls are going through a display cycle at which point the walls are opaque. A projection cycle occurs for the left eye and right eye, which yields a highly immersive experience for the user.

2.3.3 One to many eye contact

The one to many based eye contact system, created by Jones et al. (2009), is a 3D telepresence system that attempts to solve the problems of loss of eye contact and singular view rendering. The system combines a structured light based scanning system (Zhang and Huang, 2006a) along with a 360° autostereoscopic light field display (Jones et al., 2007) to create an immersive 3D experience. Although the experience is immersive 3D for the many, or the audience, it is still limited to 2D for the single user being scanned.

To achieve 3D capture of the single user, the one to many system utilizes a structured light scanner which is mounted under a 2D display that the user sits in front of. The processing pipeline decodes the structured light and computes a 640×480 depth map; the data is then transmitted across the network in a lossless format. In order to accommodate this large amount of 3D data, the system makes use of a high speed private network. A further limitation associated with the one to many eye contact system is the need to first downsample the 3D geometry to 80×60 before rendering. This is done because the 3D geometry must be rendered to seventy three unique views for the light field display, thus requiring a simple reconstruction (Jones et al., 2007). Finally, the reconstructed views are sent to the light field display which is calibrated to rotate the 3D facial geometry from the scanner coordinate system to the display coordinate system to achieve eye contact with the audience.

To capture the audience, the system uses a 2D camera along with a beam splitter to place the camera at approximately eye level with the rendered user on the light field display. This video feed is transmitted across the high speed private network and then redisplayed for the single user on a 2D display above the structured light scanner. Through the use of the beam splitter, the camera feed is retrieved from the relative position of the single users' eyes thus establishing eye contact for the single user and the audience. By achieving both sides in real-time, the system achieves eye contact in their 3D video teleconferencing system while also solving singular view rendering for the audience.

2.4 Conclusion

The goal of telepresence is to allow a person to feel as if they are present in a location other than their true location; achieving this goal is challenging as it involves a series of complex systems that can capture, transmit, and reconstruct a users senses. Realizing 3D video conferencing, a telepresence activity, involves achieving all of the telepresence stages simultaneously in real time and in 3D. This chapter examined the current research in 3D scanning, 3D compression, and 3D telepresence. While there are many different ways to perform 3D scanning, it is difficult to achieve high-resolution real-time 3D scanning, due to its computational requirements. Even if high-resolution 3D geometry can be captured in real-time, compressing it in real-time so that it may be transmitted poses another challenge. Finally, implementing all of these stages in real-time along with 3D display to achieve 3D video conferencing has proven to be nontrivial, limiting prototypes to only a few. It requires technology breakthroughs in all these fields in order to achieve 3D video conferencing while establishing correct mutual eye gaze, which is the focus of this dissertation research.

CHAPTER 3. High-resolution, real-time 3-D imaging with fringe analysis

A paper published in the Journal of Real-Time Image Processing in March 2012.[1]

Nikolaus Karpinsky[2,4] and Song Zhang[3,4]

3.1 Abstract

Real-time 3-D imaging is becoming increasingly important in areas such as medical science, entertainment, homeland security, and manufacturing. Numerous 3-D imaging techniques have been developed, but only a few of them have the potential to achieve realtime. Of these few, fringe analysis based techniques stand out, having many advantages over the rest. This paper will explain the principles behind fringe analysis based techniques, and will provide experimental results from systems using these techniques.

3.2 Introduction

With recent trends in entertainment moving towards 3-D, such as James Camerons Avatar and Radioheads House of Cards (Radiohead, 2008), a need for 3-D acquisition

[1]This paper was published in the Journal of Real-Time Image Processing and is made available as an electronic reprint with the permission of Springer. The paper can be found at the following URL on the Springer website: http://link.springer.com/article/10.1007/s11554-010-0167-4. Systematic or multiple reproduction of distribution to multiple locations via electronic or other means is prohibited and is subject to penalties under law.

[2]Primary researcher and author

[3]Assistant professor and advisor.

[4]Department of Mechanical Engineering, Iowa State University

systems is growing. Science and engineering have also begun to realize the benefits of precise 3-D measurement, yet are still looking for ways to acquire the 3-D data. Over the years, numerous techniques have been developed to acquire 3-D information such as photogrammetry, shape from focus, shape from defocus, stereo vision, spacetime stereo, moiré, speckle, holography, time of flight, interferometry, fringe projection, and structured light. Advancements in the field have allowed some of these techniques to realize real-time capability, moving into a new classification of imaging known as real-time 3-D imaging. In order to reach real-time 3-D imaging, the 3-D image acquisition, reconstruction, and display must all be realized simultaneously in real-time. Recent techniques that have been developed (Rusinkiewicz et al., 2002; Guan et al., 2003; Zhang et al., 2003; Davis et al., 2003) are based on different principals, yet rely on a common technique of using binary structured patterns. Although this binary structured pattern allows these techniques to reach real-time 3-D imaging, they share the same caveat, the spatial resolution is limited to being larger than a single projector pixel (Zhang,). For applications that require high spatial resolution this is not desirable, as the spatial resolution is limited by the projectors resolution.

To increase spatial resolution past the projectors resolution, a technique called fringe analysis may be implemented. Instead of using images with binary grayscale values, fringe analysis uses sinusoidally changing intensity values for the structured light being projected. Like binary structured patterns, more fringe images can be used to achieve higher accuracy, but this slows down the measurement speed. To reach real-time 3-D imaging, a small number of fringe images (fringe patterns) are typically used. Under certain conditions, only a single fringe pattern is needed to reconstruct the 3-D information, thus very high speed may be realized (Su and Zhang, 2010). However, as the complexity of the geometries surface increases, the measurement accuracy is affected, requiring more fringe patterns. Complex 3-D geometry requires a minimum number of three fringe images to reconstruct a single 3-D shape (Malacara, 2007). Thus, developing a real-time 3-D imaging system consists of weighing different tradeoffs, and tailoring the system to its intended application.

Throughout our research, we have developed various real-time 3-D imaging systems, tailoring each to a specific purpose. By using three fringe images, we developed a real-time 3-D imaging system that can acquire 3-D geometry at 40 frames/sec (Zhang and Huang, 2006a) with a image resolution of 532 $\times$ 500. Another system we developed measures absolute 3-D geometry at 60 frames/sec (Zhang and Yau, 2006) by using a modified 2+1 phase-shifting algorithm (Zhang and Yau, 2007b) with an image resolution of 640 $\times$ 480. Real-time 3-D geometry acquisition, reconstruction, and display has also been reached by using a multilevel quality guided phase-unwrapping algorithm (Zhang et al., 2007) and adopting the parallel processing power of a GPU (Zhang et al., 2006).

In this paper we will explain the principal behind the fringe analysis technique, summarize recent advances that use the technique, and then will present some experimental results to show the advantages and shortcomings of each technique. Since digital video projectors allow for a higher degree of flexibility over other methods, our implementations focus on producing the sinusoidal fringe patterns with a digital projector. This technique is called digital fringe projection. Although the projection method has certain unique characteristics, similar analysis approaches can be applied to fringe images generated by other techniques. To illustrate this point, we will also discuss a recently developed fringe generation technique that generates sinusoidal fringe patterns by defocusing binary structured patterns.

3.3 Digital fringe projection system

Figure 3.1 shows a typical digital fringe projection system. A computer generates a digital fringe pattern consisting of vertical stripes that are sent to a digital video projector. The projector projects the fringe images onto the object, and the objects geometry deforms the fringe images as the vertical stripes bend around the contour of the object. At this point the camera captures the distorted fringe image, and sends it to the computer for analysis. The computer analyzes the fringe image using triangulation with point correspondence, obtaining the 3-D information. Since the fringe pattern is sinusoidal, each vertical straight line corresponds to one phase value in the frequency domain. The phase value is used for 3-D reconstruction because the intensity value is very sensitive to the surface reflectivity variations. Finally, using fringe analysis the phase can be retrieved from the fringe images. The next few sections will introduce different approaches for phase retrieval.

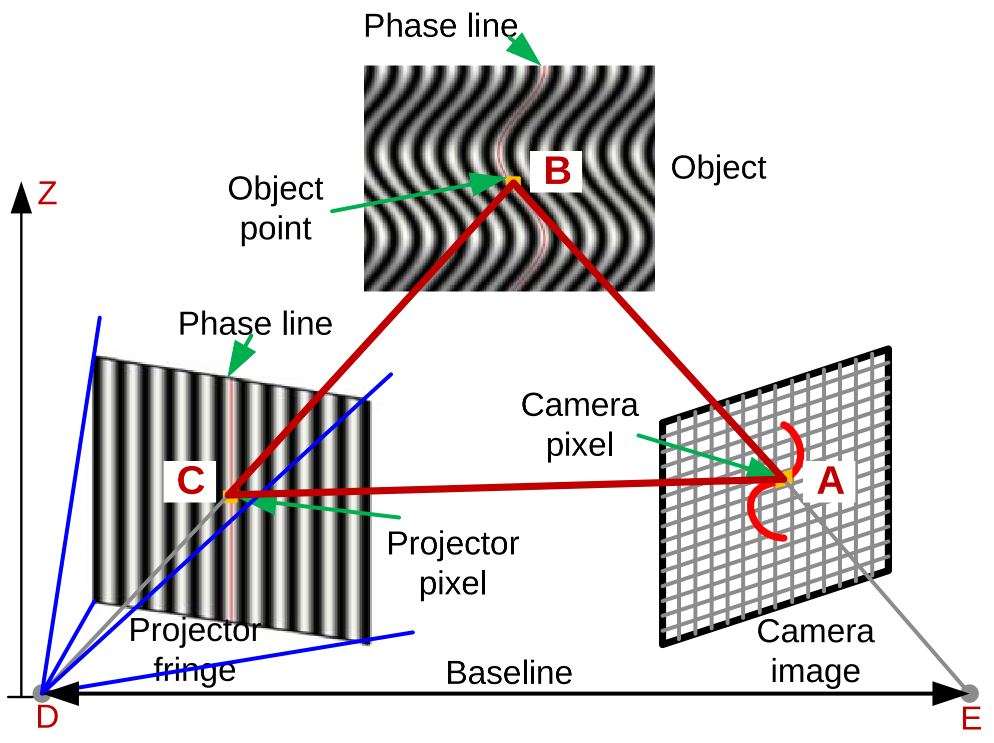


Figure 3.1 Typical setup of a digital fringe projection.

3.4 Real-time 3-D imaging using a single fringe image

If a single image is sufficient to perform 3-D shape measurement, the 3-D imaging speed can then be as fast as the image acquisition speed. This is the optimal case for real-time 3-D imaging, as 3-D reconstruction can match the camera's image acquisition speed. The technique of using a single fringe image to obtain the phase is called the Fourier method, which has been proposed by Takeda and Mutoh (1983). In this section, we will introduce a 3-D shape recovery technique using a single fringe image.

3.4.1 Principle

In general, a typical fringe image can be written as

$$I(x,y) = I'(x,y) + I''(x,y)\cos[\phi(x,y)]. \tag{3.1}$$

Here $I'(x,y)$ is the average intensity or the DC component, $I''(x,y)$ is the intensity modulation, and $\phi(x,y)$ is the phase to be solved for.

Equation (3.1) can be re-written as

$$I(x,y) = I'(x,y) + I''(x,y)\left[e^{j\phi(x,y)} + e^{-j\phi(x,y)}\right]/2. \tag{3.2}$$

In this equation, $e^{j\phi(x,y)}$ is the conjugate of $e^{-j\phi(x,y)}$. If a Fourier transform is applied, and the DC and the conjugate components are filtered out, the recovered signal by an inverse Fourier transform will give

$$\tilde{I}(x,y) = I''(x,y)e^{j\phi(x,y)}/2. \tag{3.3}$$

From Equation (3.3), the phase can be solved for by

$$\phi(x,y) = \tan^{-1}\left\{\frac{Im[\tilde{I}(x,y)]}{Re[\tilde{I}(x,y)]}\right\}. \tag{3.4}$$

Here $Im(x)$ denotes the imagery part of a complex variable x, while $Re(x)$ denotes the real part of x. The phase value ranges from $-\pi$ to $+\pi$ with 2π modus due to the arctangent. The phase map is also called the wrapped phase map. To obtain a continuous phase map, an additional step called phase unwrapping needs to be applied. The phase unwrapping step is essentially finding the 2π jumps from neighboring pixels, and removing them by adding or subtracting an integer number of 2π to the corresponding point (Ghiglia and Pritt, 1998). The 3-D coordinates can then be reconstructed using the unwrapped phase, assuming that the system is properly calibrated (Zhang and Huang, 2006b).

3.4.2 Experimental results

To illustrate how this technique works, we captured a single fringe image using a 3-D imaging system we developed. Figure 3.2(a) shows a typical fringe image captured by the camera. Figure 3.2(b) shows the image after applying the fast Fourier transform (FFT). There are two conjugate frequency components and the average.

If only a single frequency component is selected, as shown in the window, the phase can be calculated from the complex image obtained from inverse Fourier transform of the selected frequency, as shown in Figure 3.2(c). Figure 3.2(d) shows the unwrapped phase map. The unwrapped phase map can then be converted to 3-D coordinates based on the calibration of the system.

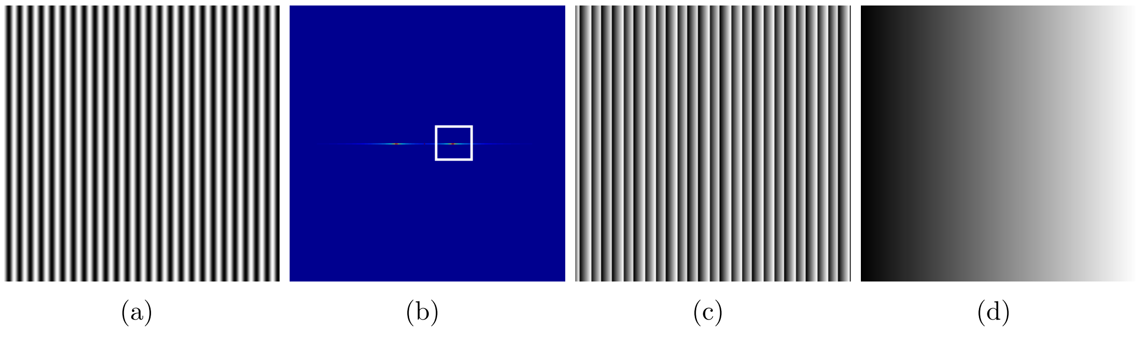


Figure 3.2 Experimental result of using a single fringe images for 3-D shape measurement. (a) Fringe image; (b) Fourier spectrum of the fringe image; (c) The phase map filtering the desired frequency signal; (d) Unwrapped phase map.

Since only one fringe image is used to obtain a 3-D shape, this technique works nicely for fast motion applications. An example of such an application is measuring vibration.

Figure 3.3 shows some frames of a vibrating cantilever beam; the fringe images are captured at 2000 fps with an exposure time of 0.5 ms. This experiment demonstrates that the 3-D profiles can be captured efficiently by this technique. This technique can also be used to measure other dynamic shapes, as surveyed in this paper (Su and Zhang, 2010).

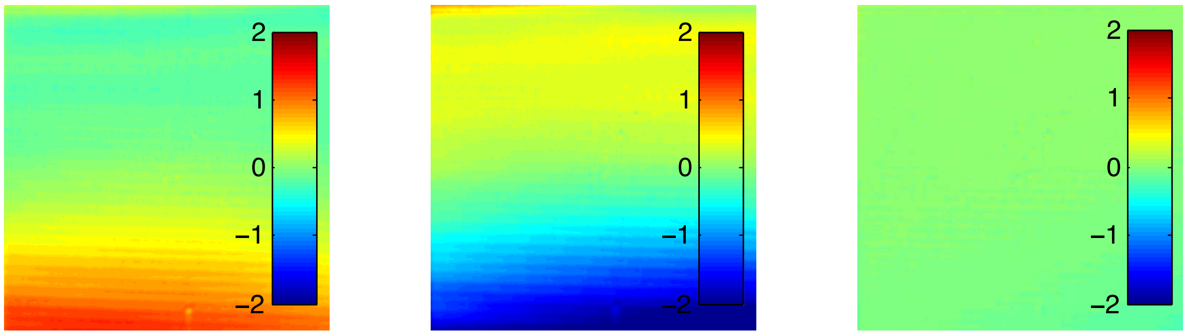


Figure 3.3 Typical measurement frames of a vibration cantilever bar. The data is captured at 2000 fps with an exposure time of 0.5 ms.

However, as illustrated in all these examples the measured surface is very smooth and uniform. This is a typical requirement for a single fringe image based real-time 3-D imaging system. This is because the single fringe technique requires the frequency of the projected fringe to be much higher than the surface geometry changes. Even with such a shortcoming, this technique is still very useful when the measurement surface meets this requirement.

3.5 Real-time 3-D imaging using two fringe images

As can be seen from the previous section, a single fringe image can be used to recover a 3-D shape. However, in order to obtain the phase in the frequency domain, the DC component must be filtered out. For a uniform fringe image with approximately uniform surface reflectivity, the DC frequency only covers a very small area, thus can be easily filtered out. However, if the surface reflectivity varies significantly across the surface, it would be challenging to separate the DC component and the desired frequency. To alleviate this problem, Guo and Huang (2008) introduced a technique that uses two fringe images.

3.5.1 Principle

The principle of this technique is similar to the previously introduced method except it requires the capturing of an additional image

$$I_a(x, y) = I'(x, y). \tag{3.5}$$

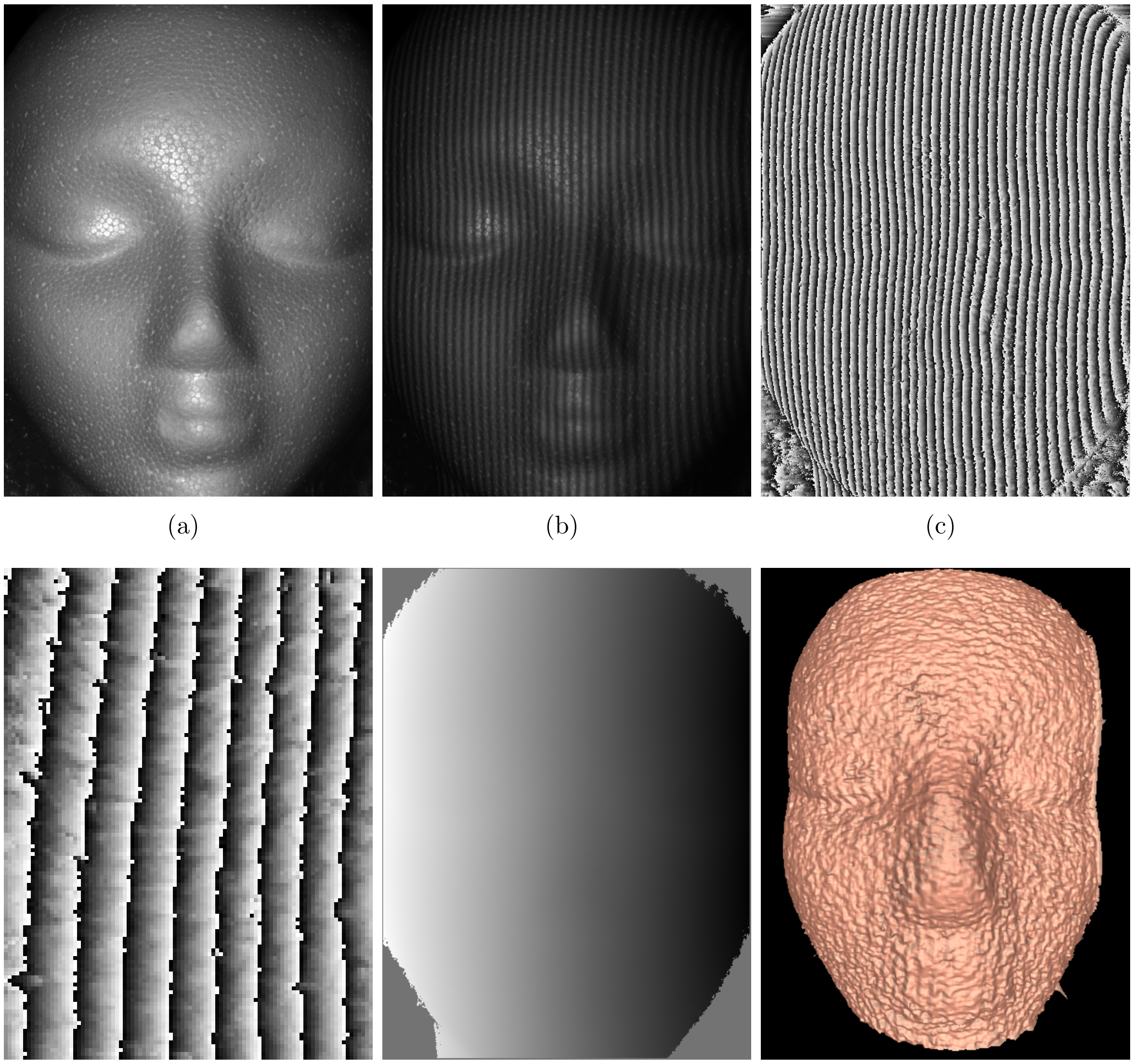
Before applying the Fourier transform, the fringe image is obtained by taking the difference between the images.

$$I_m(x, y) = I(x, y) - I_a(x, y) = I''(x, y)\cos[\phi(x, y)]. \tag{3.6}$$

By doing this, the DC component will not affect the measurement significantly, thus the measurement quality can be improved. However, it sacrifices measurement speed as two fringe images must be captured to construct a 3-D shape.

3.5.2 Experimental results

To verify the performance of this technique, we compared it to the single fringe technique, measuring a sculpture with non uniform surface reflectivity as shown in Figure 3.4(a). The captured fringe is shown in Figure 3.4(b). The figure clearly shows that the fringe intensity varies from point to point. Figure 3.4(c) shows the phase map, and Figure 3.4(d) shows the zoom in view of the phase map. There are significant artifacts in the phase map which will cause measurement error. The unwrapped phase map is shown in Figure 3.4(e), and the reconstructed 3-D shape is shown in Figure 3.4(f). This experiment shows that if the surface is complex, a single fringe image cannot produce high quality 3-D measurement.



(a) (b) (c) (d) (e) (f)

Figure 3.4 More complex 3-D surface measurement result using a single fringe image. (a) Photograph of the object to be measured; (b) Fringe image; (c) Wrapped phase map; (d) Zoom-in view of the unwrapped phase map; (e) unwrapped phase map; (f) 3-D reconstructed shape rendered in shaded mode.

If the average image $I_a(x, y)$ (shown in Figure 3.5(a)) is captured and subtracted from the fringe image (as shown in Figure 3.4(b)), the phase map can be obtained with higher measurement quality. Figure 3.5(b) shows the zoom-in view of the phase image, clearly showing the improvement of the phase map. Thus the 3-D measurement will be significantly enhanced. Figure 3.5(c) shows the 3-D shape. By using two fringe images, the influence of surface reflectivity variations is significantly reduced, thus improving the measurement quality. With this technique, a lot more detail can be recovered.

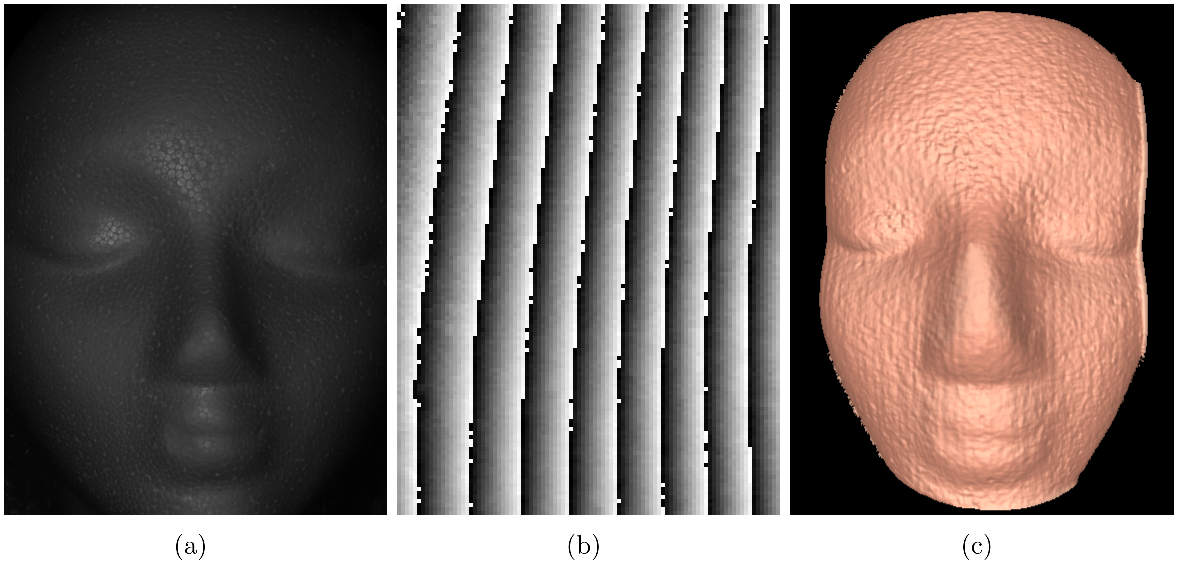


Figure 3.5 More complex 3-D shape using two fringe images. (a) The average image $I_a(x, y)$; (b) Zoom in view of the phase map; (c) 3-D reconstructed result.

Even though the measurement quality is significantly improved, it is still very difficult for this technique to measure complex 3-D surfaces because of its fundamental limitation: surface changes must be slower than the fringe changes. This is because the phase cannot be solved for explicitly from the fringe image point by point. If the phase can be solved for each individual point, the surface structure requirement can be eliminated since it would not rely on the neighboring pixel information to retrieve the phase. Equation (3.1) shows that there are three unknowns in the equation $I'(x, y)$, $I''(x, y)$, and $\phi(x, y)$. Therefore, if another image is provided, the phase $\phi(x, y)$ can be uniquely solved. This is where the requirement for three fringe images for a single 3-D shape measurement comes from.

3.6 Real-time 3-D imaging using three fringe images

From three fringe images the phase can be uniquely solved for, thus it allows for the measurement of complex 3-D shapes. In this section, we will introduce a real-time 3-D imaging system based on this technique.

3.6.1 Principle

3.6.1.1 Three-step phase-shifting algorithm

The previously introduced methods depended on obtaining the phase from Fourier analysis. If three fringe images are available, the phase can be uniquely solved. Among these multiple fringe analysis techniques, the technique based on phase-shifting has been widely adopted in optical metrology (Malacara, 2007). In this technique, the fringe is shifted spatially from frame to frame with a known phase shift. By analyzing a set of phase-shifted fringe images, the phase can be obtained. The phase-shifting based techniques have the following advantages: (1) high measurement speed, because it only requires three fringe images to recover one 3-D shape; (2) high spatial resolution, because the phase can be obtained pixel by pixel, thus the measurement can be performed pixel by pixel; (3) less sensitivity to surface reflectivity variations, since the calculation of the phase will automatically cancel out the DC components.

Over the years, a number of phase-shifting techniques have been developed, including three-step, four-step, double three-step phase shifting and others. For real-time 3-D imaging, using the minimum number of fringe images is desirable, thus a three-step phase-shifting algorithm is usually used. For a three-step phase-shifting algorithm, if the phase shift is $2\pi/3$, the three required phase-shifted fringe images can be written as

$$I_1(x,y) = I'(x,y) + I''(x,y)\cos[\phi(x,y) - 2\pi/3], \quad (3.7)$$

$$I_2(x,y) = I'(x,y) + I''(x,y)\cos[\phi(x,y)], \quad (3.8)$$

$$I_3(x,y) = I'(x,y) + I''(x,y)\cos[\phi(x,y) + 2\pi/3]. \quad (3.9)$$

Solving Eqs.(3.7)–(3.9) simultaneously, we obtain the average intensity

$$I'(x,y) = (I_1 + I_2 + I_3)/3, \quad (3.10)$$

the intensity modulation

$$I''(x,y) = \frac{\sqrt{3(I_1 - I_3)^2 + (2I_2 - I_1 - I_3)^2}}{3}, \quad (3.11)$$

the phase

$$\phi(x,y) = \tan^{-}1\left[\frac{\sqrt{3}(I_1 - I_3)}{2I_2 - I_1 - I_3}\right]. \quad (3.12)$$

From the same equations, we can also obtain the texture image, $I_t(x,y) = I'(x,y) + I''(x,y)$, and the data modulation

$$\gamma(x,y) = \frac{I''}{I'} = \frac{\sqrt{3(I_1 - I_3)^2 + (2I_2 - I_1 - I_3)^2}}{I_1 + I_2 + I_3}. \quad (3.13)$$

The data modulation indicates the fringe quality, with 1 being the best. Data modulation is often valuable for phase unwrapping where the progress path is vital (Ghiglia and Pritt, 1998). A measurement example using the three fringe images is shown in Fig. 3.6. It clearly shows that the measurement quality is much better than that of a two fringe image or a single fringe image based technique.

3.6.1.2 Real-time 3-D image acquisition

Since only three fringe images are used, they can be encoded as the three primary color channels (red, green and blue, or RGB) and projected at once (Pan et al., 2006; Geng, 1996; Huang et al., 1999). However, the measured quality is affected to a various

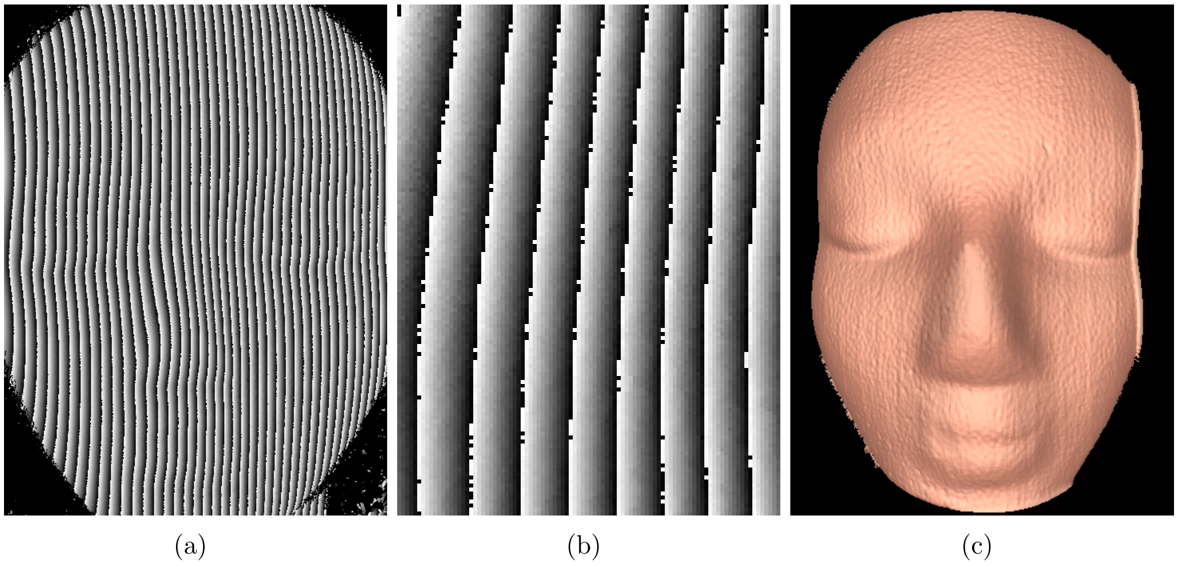


Figure 3.6 More complex 3-D shape using three fringe images. (a) The phase map; (b) Zoom in view of the phase map; (c) 3-D reconstructed result.

degree if the measured surface has strong color variations. In addition, the color coupling between RG and GB also affect the measurement quality if no filtering is used (Pan et al., 2006).

The problems induced by using color fringe patterns can be eliminated if three monochromatic fringe patterns are rapidly projected sequentially. The unique projection mechanism of a single-chip digital-light-processing (DLP) projection system makes this rapid switching feasible, thus real-time 3-D imaging can be realized. Figure 3.7 shows the layout of such a system. Three phase-shifted fringe images are encoded as the RGB channels of the projector. The color fringe image is then projected by the projector channel by channel sequentially. If the color filters of the DLP projector are removed, the color images will be be projected in monochromatic mode, thus the problems related to color are not present. A high-speed CCD camera, synchronized with the projector is used to capture the three channel images one by one. By applying the three-step phase-shifting algorithm to three fringe images, the phase and thus the 3-D shape can be obtained. By this means, we have reached 3-D data acquisition at 40 fps (Zhang and

Huang, 2006a), and later on achieved 60 fps (Zhang and Yau, 2007b).

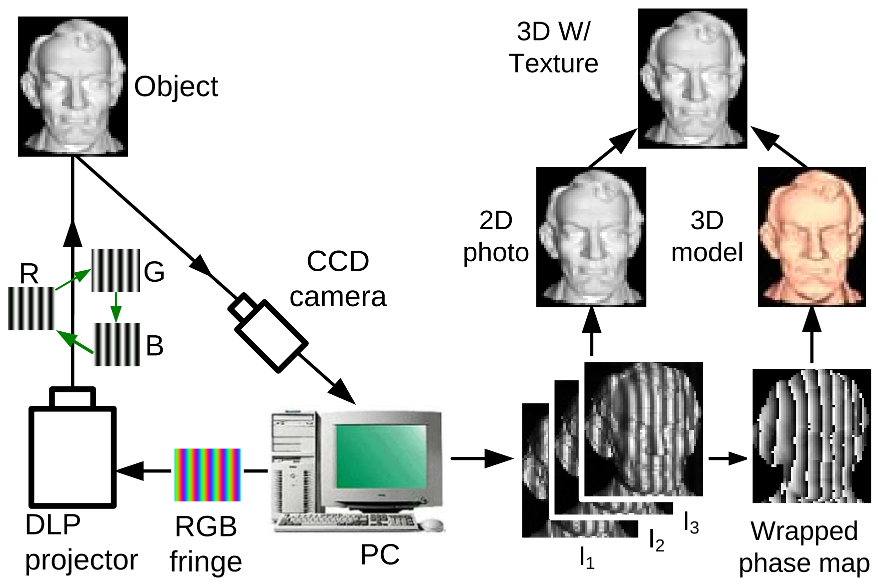


Figure 3.7 The layout of the real-time 3-D imaging system we developed.

3.6.1.3 Real-time 3-D reconstruction and visualization

In order to realize real-time 3-D imaging, the phase wrapping and unwrapping speed must be realized in real-time. We have developed a fast three-step phase-shifting algorithm to improve the phase wrapping speed by about 3.4 times (Huang and Zhang, 2006). This algorithm essentially approximates the arctangent function with an intensity ratio calculation in the same manner as that of the trapezoidal phase-shifting algorithm (Huang et al., 2005). The approximation error is then compensated for by a small look-up-table (LUT).

The phase unwrapping obtains the smooth phase map by removing the 2π discontinuities. Over the years, numerous robust phase unwrapping algorithms have been developed that include the branch-cut algorithms (Huntley, 1989; Salfity et al., 2006), the discontinuity minimization algorithm (Flynn, 1997), the L^p-norm algorithm (Ghiglia and Romero, 1996), the region growing algorithm (Baldi, 2003; Hung and Yamada, 1998),

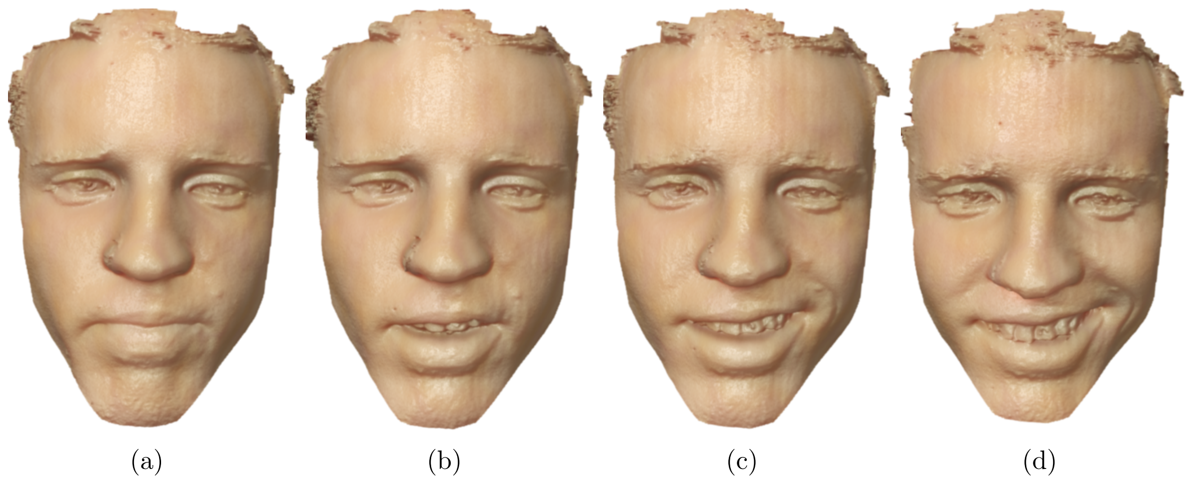


Figure 3.8 Facial shape measurement. (a) Shows subject in neutral state; (b) - (d) Show expression on subjects face evolving through time into a smiling state. The data is captured at 60 fps with a resolution of 640 × 480.

the agglomerative clustering-based approach (Merráez et al., 2005), and the least-squares algorithms (Chyou et al., 2004). However, a conventional robust phase unwrapping is usually very slow for real-time processing: it takes anywhere from a few seconds, to a few minutes, to even a few hours to process a standard 640 × 480 phase map (Ghiglia and Pritt, 1998). There are some rapid but not robust phase unwrapping algorithms, such as the flood filling and the scan line algorithms (Zhang and Huang, 2006a; Zhang et al., 2007). However, these algorithms are usually very sensitive to the noise of the phase map.

By combining the advantages of the the rapid scan line phase shifting algorithm and a robust quality guided phase-shifting algorithm, we developed a real-time phase unwrapping algorithm (Zhang et al., 2007). In particular, the phase map is evaluated and quantified into different quality levels: the unwrapping process starts with the highest quality level points using the scan line algorithm and then moves to the lower quality ones. With this method, the lower quality points will not propagate and drastically affect the high quality points. This algorithm is a tradeoff between robustness and speed. We are mostly interested in human facial expression measurement, and for this application this algorithm works reasonably well. Figure 3.9 shows a typical measurement result of human facial expressions. It should be noted that because of the facial hair, the surface reflectivity variations are large, and the phase unwrapping is usually very challenging, but our algorithm can unwrap the phase correctly for most cases.

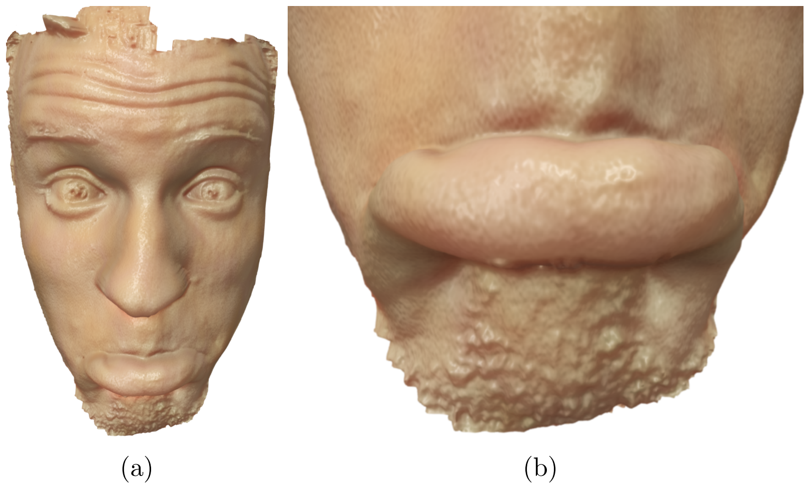


Figure 3.9 Facial shape measurement. (a) 3-D facial data rendered in shaded mode; (b) Zoom-in view of the mouth region. The data is captured at 60 fps with a resolution of 640 $\times$ 480.

To reach real-time 3-D imaging, the phase map has to be converted to 3-D coordinates and visualized in real-time. We found that it is very challenging to accomplish these tasks by central processing units (CPUs). Because the phase-to-coordinate conversion consists of simple point by point matrix operations, it can be efficiently calculated in parallel on a graphics processing unit (GPU). Because the input data is just the phase value at each point instead of computed 3-D coordinates, data transfer from the CPU to the GPU is drastically reduced. In addition, because the coordinates are computed on GPU, they can be rendered immediately without accessing CPU data. By adopting this technique, real-time real-time 3-D imaging has been accomplished by using a three-step phase-shifting algorithm (Zhang et al., 2006).

3.6.2 Experimental results

Figure 3.10 shows the developed hardware system. The whole size of the system is approximately 15″ × 12″ × 14″. The system uses a high-speed CCD camera (Pulnix TM-6740 CL), with a maximum frame rate of 200 fps. The sensor pixel size is H: 7.4 μm and V: 7.4 μm. The maximum data speed for this camera is 200 frames per second. The camera resolution is 640 × 480, and the lens used is a Fujinon HF16HA-1B f = 16 mm lens. The projector (PLUS U5-632h) has an image resolution of 1024 × 768, and a focal length of f = 18.4–22.1 mm. This projector refreshes at 120 Hz, thus, this system can theoretically reach 120 Hz 3-D imaging speed. However, due to the speed limit of the camera, we can only capture fringe images at 180 fps. Since three fringe images are needed to reconstruct one 3-D shape, the 3-D imaging speed is actually 60 Hz.

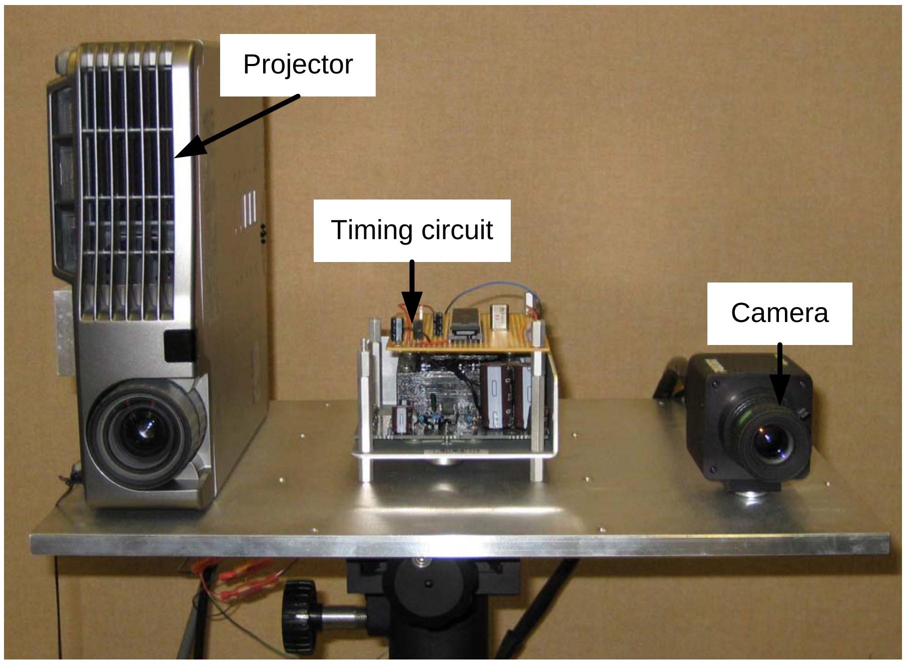


Figure 3.10 The photograph of the real-time 3-D imaging system.

High quality natural facial expressions can be captured since the measurement speed is so fast. Figure 3.8 shows frames from the forming of a human facial expression. It clearly indicated that the details of the facial expression are captured fairly well. It should be noted that the data was processed with a 7×7 Gaussian smoothing filter to reduce the most significant random noise.

As introduced earlier, we are not only able to acquire 3-D shape in real-time, but also are able to simultaneously process and display them at high speed. Figure 3.11 shows an experimental result of measuring human face. The right figure shows the real subject, and the left shows the 3-D geometry acquired and rendered on the computer screen at the same time. The simultaneous 3-D data acquisition, reconstruction, and display speed achieved is 30 frames/sec.

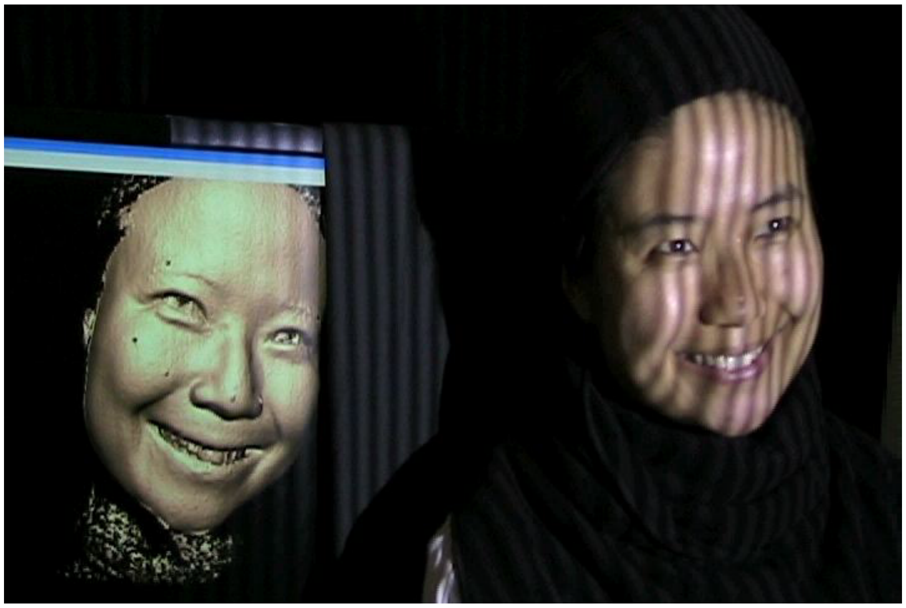


Figure 3.11 Simultaneous 3-D data acquisition, reconstruction, and display. The system achieved a real-time 3-D imaging at a frame rate of 30 fps with an image resolution of 640×480 per frame.

3.7 New fringe generation technique

Conventionally, fringe images are either generated by laser interference, or by a fringe projection technique. A fringe projection technique is broadly used because of its flexibility in generating sinusoidal fringe patterns. However, it usually requires at least 8-bit grayscale values to represent a high-quality fringe image. Because of the complexity of a fringe projection system, it usually has the following shortcomings:

- *Projector nonlinearity problem.* Because the projector is a nonlinear device, generation of sinusoidal fringe images is difficult. Different algorithms have been proposed to calibrate and correct the nonlinearity of the projector (Huang et al., 2002; Kakunai et al., 1999; Zhang and Huang, 2007; Zhang and Yau, 2007a; Guo et al., 2004; Pan et al., 2009), but it remains difficult to accurately represent the nonlinear curve.To some extent, the nonlinearity of the system is also caused by the graphics card that connects with the projector. We found that the nonlinearity of the projector needs to be re-calibrated if a different video card is used. We also found that the projector's nonlinearity actually changes over time, thus frequent calibration is needed for high quality measurement. This is certainly not desirable because the nonlinearity calibration is usually a time consuming procedure.

- *Synchronization problem.* Since a DLP projector is a digital device, and it generates images by time integration (Hornbeck, 1997), the synchronization between the projector and the camera is vital. Any mismatch will results in significant measurement error (Lei and Zhang, 2010).

- *Minimum exposure time limitation.* Because the projector and the camera must be precisely synchronized, the minimum exposure time used for each fringe image capture is actually limited by the projector's refresh rate, typically 120 Hz. Thus, the minimum exposure time to use is actually 2.78 ms. However, when capturing fast motion, a shorter exposure time is usually required. This technique limits potential fast motion capture applications.

All these above mentioned issues are introduced by the conventional fringe generation technique, i.e., using 8-bit grayscale images. If a new fringe generation mechanism is employed that only requires 1-bit images, all these problems can be either avoided or significantly reduced. This is the motivation to explore this potential possibility.

3.7.1 Principle of fringe pattern generation using defocusing

We recently found that by defocusing binary structured patterns, sinusoidal fringe patterns can be generated (Lei and Zhang, 2009). This is based on our two observations: (1) seemingly sinusoidal fringe patterns often appear on the ground when the light shines through an open window blinds; (2) the sharp features of an object are blended together in a blurring image that is captured by an out-of-focus camera. The former gives the insight that an ideal sinusoidal fringe pattern could be produced from a binary structured pattern; the latter provides the hint that if the projector is defocused, the binary structured pattern might become an ideal sinusoidal one.

Figure 3.12 illustrates how to generate sinusoidal fringe patterns by using defocusing. If the projector is defocused to different degrees, the binary structured patterns are deformed differently. When the projector and the camera are in focus, the fringe patterns have very obvious binary structures, as shown in Fig. 3.12(a). With the increase of the defocusing degree, the binary structures are less and less clear, and they become more and more sinusoidal. However, if the defocusing degree is too much the sinusoidal structure becomes obscure, as indicated in Fig. 3.12(f). It should be noted that during all experiments, the camera is always in focus.

This technique has the following advantages compared with the conventional fringe generation techniques (Lei and Zhang, 2010):

1. There is no need to calibrate the nonlinear gamma of the projector because only two intensity levels are used;

2. It is very easy to generate sinusoidal fringe patterns because no complicated algorithms are necessary;

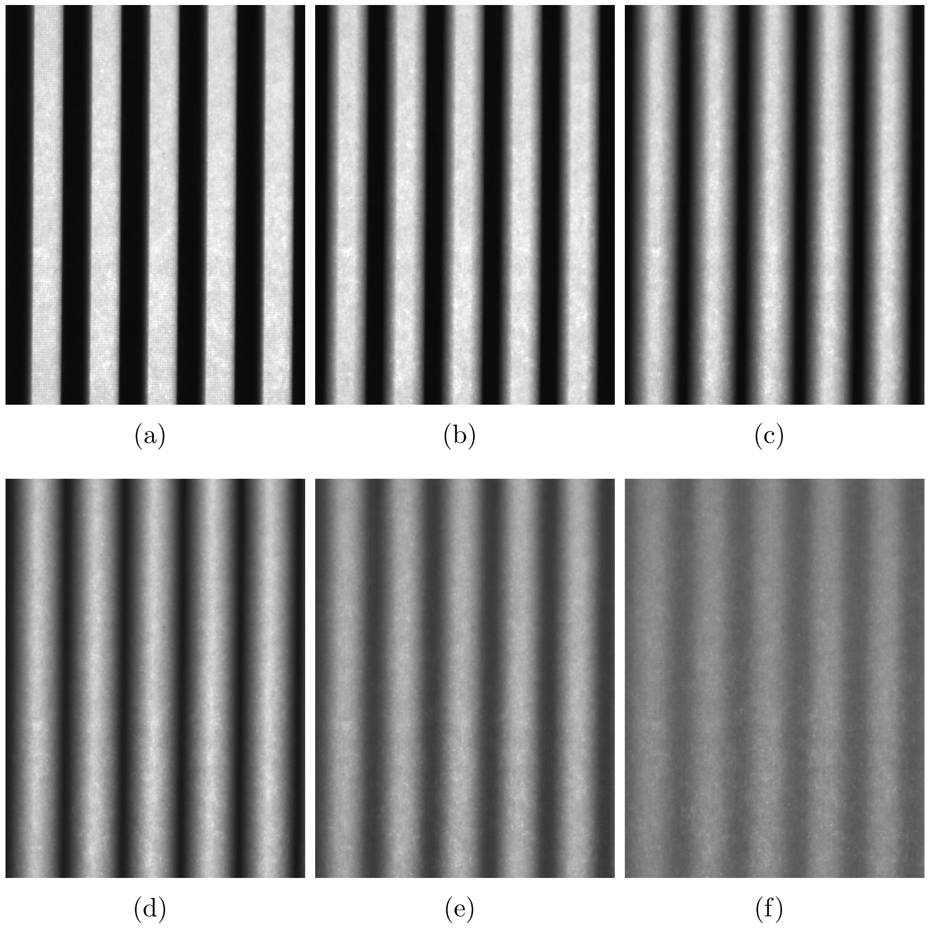


Figure 3.12 Example of sinusoidal fringe generation by defocusing a binary structured patterns. (a)-(f) shows when the projector is defocused at different degrees, (a) shows the projector is in focus, while (f) shows that when the projector are defocused too much.

3. There is no need to precisely synchronize the projector with the camera; and

4. The measurement is less sensitive to exposure time used. Therefore, the defocused binary pattern method is advantageous for 3-D shape measurement using a commercial DLP projector.

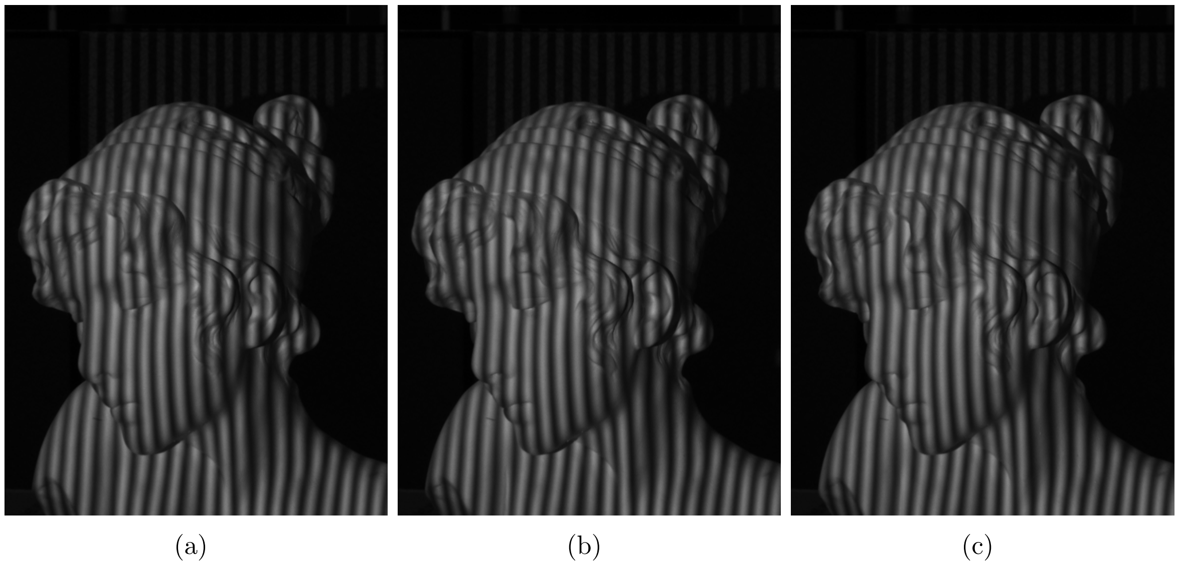
3.7.2 Experimental results

This technique has been verified by measuring a complex sculpture, as shown in Fig. 3.13. Figure 3.13(a)-3.13(c) show three phase shifted fringe images with a phase shift of $2\pi/3$. The phase shifting is realized by spatially moving the binary structured patterns. For example, $2\pi/3$ phase shift is realized by moving the structured patterns 1/3 of the period. These fringe images can then be analyzed by the three-step phase-shifting to perform the measurement. It clearly shows that the measurement quality is very high.

Due to the numerous advantages of using this technique to generate sinusoidal fringe patterns, it has the potential to replace the conventional fringe generation technique for 3-D imaging with fringe analysis techniques.

3.8 Conclusion

This paper has presented techniques for real-time 3-D imaging, introducing the theory, and providing experimental results demonstrating the capabilities of each. With the Fourier analysis technique, a single fringe image can be used to reconstruct one 3-D shape, although it requires surface uniformity. Essentially, this technique can reach pixel level resolution with 3-D reconstruction speed equal to the frame rate of the camera. Geometry that has surface height variations or reflectivity changes with a frequency above the fringe frequency used cannot be captured with a single fringe image; in such cases, dual fringe images may be used. Using dual fringe images improves measurement



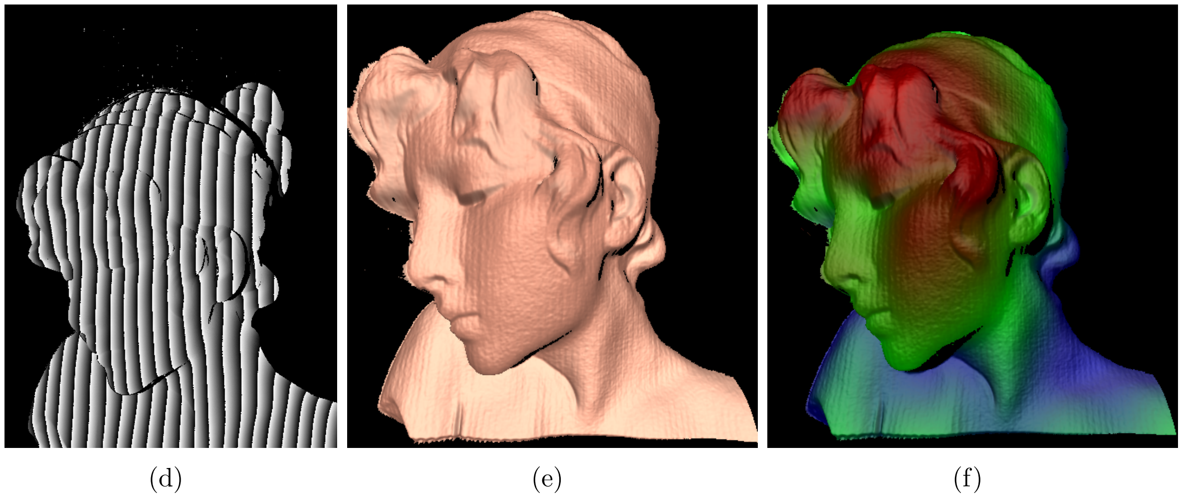


Figure 3.13 Example of sinusoidal fringe generation by defocusing a binary structured patterns. (a) $I_1(-2\pi/3)$k; (b) $I_2(0)$; (c) $I_3(2\pi/3)$; (d) Wrapped phase map; (e) 3-D reconstructed result; (f) 3-D shape with color encoded depth map.

accuracy, but 3-D reconstruction speed is reduced to half the frame rate of the camera. Furthermore, three fringe images can be used, along with a phase-shifting technique; significantly improving accuracy, but reducing the frame rate to a third of the frame rate of the camera. With a camera that has a frame rate above 90 frames per second, we developed a real-time 3-D imaging system that can simultaneously acquire, reconstruct, and display 3-D geometry at 30fps at a resolution of over 300,000 points per frame. Due to the very challenging nature of generating ideal sinusoidal fringe patterns, a novel approach was also discussed to significantly simplify the fringe generation, which shows great potential to be the mainstream in this field.

3.9 Acknowledgements

The authors would like to thank William Lohry for his contribution on rendering the facial data with Blender, and Victor Villagomez and Ying Xu for processing the data. All these individuals are undergraduate students working under the supervision of Dr. Zhang in the 3-D Machine Vision Laboratory at Iowa State University.

CHAPTER 4. High-Speed 3D Scanning on a Portable Device

A paper submitted to the *Journal of Optical Engineering* in October 2013.[1]

Nikolaus Karpinsky[2,6], Morgan Hoke[3,6], Vincent Chen[4,6], and Song Zhang[5,6]

4.1 Abstract

This paper presents a novel 3D shape measurement system that can simultaneous achieve 3D shape acquisition, reconstruction, and display at 30 frames per second (fps) with 480,000 measurement points per frame. The entire processing pipeline was realized on a graphics processing unit (GPU) without the need of substantial central processing unit (CPU) power, making it achievable on a portable device, namely a laptop computer. Furthermore, the system is extremely inexpensive compared with similar state-of-art systems, making it possible to be accessed by the general public. Specifically, advanced GPU techniques such as multipass rendering and offscreen rendering were used in conjunction with direct memory access to achieve the aforementioned performance. This paper will present the developed system, implementation details, and experimental results to verity the performance of the proposed technique.

[1]This paper was submitted in the Journal of Optical Engineering Engineering. Systematic or multiple reproduction of distribution to multiple locations via electronic or other means is prohibited and is subject to penalties under law.

[2]Primary researcher and author

[3]Undergraduate student collaborator

[4]Undergraduate student collaborator

[5]Assistant Professor and advisor.

[6]Department of Mechanical Engineering, Iowa State University

4.2 Introduction

Recent advances in technology have enabled high-resolution, real-time 3D shape measurement through the use of structured light techniques (Zhang et al., 2006; Liu et al., 2010). (Note that real-time 3D shape measurement includes 3D shape acquisition, reconstruction, and display all in real time). While these advances are impressive, they require large amounts of computing power, thus being limited to using large desktop workstations with high end central processing units (CPUs) and sometimes graphics processing units (GPUs). This is undesirable in making high-resolution real-time 3D scanners ubiquitous in our mobile lives due to their costs and portability issues. Recently, advancements in the speed of structured light based techniques have pushed them into real-time without being computationally intensive and are more portable; this growth into real-time has given structured light scanners broad exposure, as can be seen with the introduction of the Microsoft Kinect. Though successful, the Kinect only provides sparse data points with low accuracy. Achieving high-resolution, real-time 3D shape measurement on a low-end computer, such as a portable laptop still remains challenging.

Similar to a stereo vision technique, structured light scanning works off of the principle of triangulation. Instead of using two cameras, such as the case of stereo vision, structured light scanning replaces a camera with a projector (Salvi et al., 2004). The projector projects a set of encoded patterns, which are used to establish a correlation between the projector and camera images, thus circumventing the correlation problem in stereo imaging. Assuming the system is calibrated, 3D information can be triangulated using the established correspondence (Zhang and Huang, 2006b).

Decoding patterns and performing triangulation is a computationally intensive task, making it difficult to reach real-time speeds using serial processing methods, as is seen with traditional CPUs. If the processes can be realized utilizing parallel algorithms, parallel compute devices such as a graphics processing unit (GPU) can be used to offload the computationally intensive problem (Ahrenberg et al., 2009; Espinosa-Romero and Legarda-Saenz, 2011; Gao and Kemao, 2010; Carpenter and Wilkinson, 2010; Kang et al., 2008). Although the clock speed on a GPU is not as fast as a CPU, anywhere from 1-8 times slower, it can process hundreds of threads simultaneously, assuming there is no branching (Asano et al., 2009). Gao and Kemao provide a good review on parallel compute devices and their application in optical measurement (Gao and Kemao, 2012). If the GPU can be leveraged on a portable device, then it has the potential to reach real-time 3D shape measurement speeds even with the devices limited computational power.

To establish the correlation between projector and camera pixels, the encoded patterns must be properly chosen to allow for parallel computation, maximum speed, yet being resilient to noise from things such as ambient light, surface reflections, etc. Many different techniques such as stripe boundary code (Rusinkiewicz et al., 2002), binary coded patterns (Ishii et al., 2007), and phase shifting methods (Zhang and Yau, 2006) exist. Although codification strategies such as binary coded patterns are parallel in nature, thus being well suited for GPU implementation, they typically require many patterns and are limited to the number of projector pixels. Fringe projection on the other hand uses sinusoidally varying fringe images which no longer limits the measurement to the number of projector pixels, but requires spatial phase unwrapping (Zhang, 2010). Since phase unwrapping is typically a serial operation, it is not well suited to parallel implementation and difficult to achieve on a GPU (Gao and Kemao, 2010).

To address this issue, this work describes and demonstrates a real-time 3D shape measurement system that is realized on a portable device, namely a laptop computer, which achieves a speed of 30 frames per second (fps) at an image resolution of 800×600. This technique is based on a modified two-frequency phase-shifting technique with binary defocusing (Lei and Zhang, 2009; Wang and Zhang, 2011). The two-frequency phase-shifting algorithm enables pixel-by-pixel phase computation yet only requires a small number of fringe patterns (only 6); the proposed, modified two-frequency phase-shifting algorithm, is also less sensitive to the noise caused by the two-frequency phase-shifting algorithm. To achieve high-resolution real-time 3D shape measurement on a portable device, our approach utilizes a GPU as a multipurpose parallel co-processor. Through the use of the OpenGL Shading Language (GLSL) we have created a structured light processing pipeline that is implemented solely in parallel on the GPU. This reduces the processing power requirements of the device performing 3D reconstruction, allowing us to realize the system with a portable device, namely a laptop computer. To mitigate high speed camera transfer problems, which typically require a dedicated frame grabber, we make use of USB 3.0 along with direct memory access (DMA) to transfer camera images to the GPU without the need of synchronization between the CPU and GPU. In addition to low processing power requirements, since the entire system is realized on the GPU, the CPU is nearly ideal, thus freeing it perform other operations in parallel to the 3D reconstruction, such as 3D registration or feature extraction. We developed a low-cost system (less than $3,000 including the laptop computer) that can achieve 30 fps measurement speed with 480,000 points per frame.

Section 4.3 of this paper explains the principles of the techniques employed by the system. Section 4.4 breaks the implementation of the system down into stages and discusses how each stage is achieved on the GPU. Section 4.5 shows the experimental results to demonstrate the success of the proposed techniques. Finally, Section 4.6 summarizes the paper.

4.3 Principle

4.3.1 Two-frequency phase-shifting algorithm

The structural patterns used by our system are a set of three-step phase-shifted fringe patterns. A phase shifting method was chosen over other structured light coding techniques due to it only requiring a few patterns for codification resulting in high speed, it not being limited by projector resolution, and its resilience to noise (Zhang, 2010). Three-step phase-shifted patterns can be described by

$$I_1(x,y) = I'(x,y) + I''(x,y)\cos[\phi(x,y) - 2\pi/3], \tag{4.1}$$

$$I_2(x,y) = I'(x,y) + I''(x,y)\cos[\phi(x,y)], \tag{4.2}$$

$$I_3(x,y) = I'(x,y) + I''(x,y)\cos[\phi(x,y) + 2\pi/3]. \tag{4.3}$$

Here I' is the average intensity, I'' is the intensity modulation, and ϕ is the encoded phase. Using these equations, the phase ϕ can be solved by

$$\phi(x,y) = \tan^{-1}\left[\frac{\sqrt{3}(I_1 - I_3)}{2I_2 - I_1 - I_3}\right]. \tag{4.4}$$

This equation yields a phase value ϕ for every pixel, but since the $\tan^{-1}$ only ranges from 0 to 2π, the phase value provided will have 2π phase discontinuities; this phase value is known as wrapped phase. Conventional approaches employ a spatial phase unwrapping algorithm that traverses along the phase map adding multiples of 2π (Ghiglia and Pritt, 1998), but this is a serial operation that requires neighboring pixel information thus being undesirable for GPU implementation.

Instead, we adopted a two-frequency phase-shifting algorithm to temporally unwrap the phase pixel by pixel, which is well suited for GPU implementation. For the temporal phase unwrapping algorithm, two frequencies are the minimum number needed to unwrap, although more can be utilized for higher quality phase; to achieve real-time speeds, our implementation chose only two, thus requiring six fringe images. Briefly, the two-frequency phase-shifting algorithm works as follows: we obtain the phase map ϕ_1 from one set of phase-shifted fringe patterns with frequency of f_1 (or fringe period of T_1), and phase map ϕ_2 from the second set of phase-shifted fringe patterns with a different frequency of f_2 (or fringe period of T_2). Instead of using fringe frequencies or fringe periods, conventionally, the *wavelength* was used because such an algorithm was developed for laser interferometers where the wavelength λ of the laser light has a physical meaning, and can uniquely determine the interfered fringe pattern. However, in a digital fringe projection (DFP) system, the fringe patterns are directly generated by a computer, and the wavelength of light used does not have any correlation with the fringe patterns generated. Therefore, in this paper, we use fringe period or fringe frequency instead. The fringe period is defined as the number of pixels per period of the fringe.

By utilizing the phase difference of the phases ϕ_1 and ϕ_2 with different fringe frequencies and the modus operation, an equivalent phase map can be obtained

$$\phi_{eq} = \phi_1 - \phi_2 \bmod 2\pi. \tag{4.5}$$

This resultant phase map has a different fringe period from T_1 and T_2, which is usually called equivalent fringe period, T_{eq}, that can be determined by

$$T_{eq} = \frac{T_1 T_2}{\mid T_1 - T_2 \mid}. \tag{4.6}$$

By properly selecting the spatial frequencies of the phases ϕ_1 and ϕ_2, a continuous phase ϕ_{eq}, that spans the entire phase map without 2π discontinuities can be achieved. The equivalent phase map ϕ_{eq} can then be utilized to unwrap the phase ϕ_1 or ϕ_2 point by point by

$$k(x,y) = \text{round}\left(\frac{\phi_{eq}(x,y) \times T_{eq}/T_1 - \phi_1(x,y)}{2\pi}\right), \tag{4.7}$$

$$\Phi(x,y) = \phi_1(x,y) + 2\pi \times k(x,y). \tag{4.8}$$

This two-frequency phase-shifting approach can obtain an unwrapped phase map Φ per pixel in parallel, thus it is well suited for GPU implementation. Figure 4.1 shows a set of captured fringe images with two fringe periods, $T_1 = 60$ and $T_2 = 66$ pixels. Their corresponding wrapped and unwrapped phase maps are shown in Figs. 4.1(c)-4.1(f).

4.3.2 General purpose GPU

Recently, in order to accelerate parallel tasks on a computer, general purpose GPU (GPGPU) computation has been leveraged. The main goal of GPGPU computation is

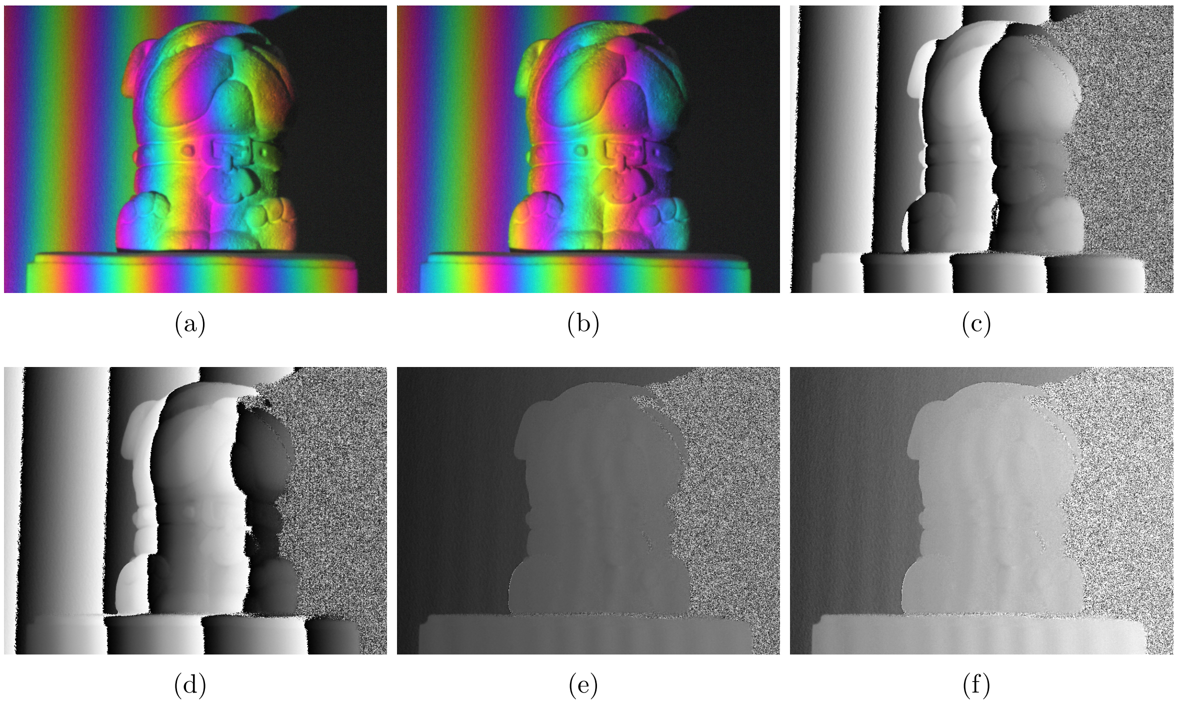


Figure 4.1 Two-frequency phase unwrapping process. (a) three fringe images of smaller phase; (b) three fringe images of larger phase; (c) wrapped phase ϕ_1; (d) wrapped phase ϕ_2; (e) equivalent phase ϕ_{eq}; (f) unwrapped phase Φ for ϕ_1.

to free the CPU of a computer from parallel intensive tasks by leveraging the GPU as a parallel co-processor (Gao and Kemao, 2010). Although not having nearly as high a clock speed as modern CPUs, GPUs have many more processing cores, typically on the scale of hundreds to thousands. To leverage this technology in applications, such as 3D reconstruction, different programming interfaces can be used such as NVIDIA CUDA (CUD, 2007), OpenCL (Munshi et al., 2009), or the OpenGL Shading Language (GLSL) (Rost et al., 2005).

While GPGPU programming interfaces such as CUDA and OpenCL offer lower level hardware access to features such as shared memory, they are only beginning to be supported in portable and mobile devices with platforms such as the NVIDIA Tegra. Conversely, GLSL is supported on nearly all modern graphics devices, is part of the OpenGL ES specification (Munshi, 2007), and is better for interoperability with a graphics API (Fang et al., 2011). In terms of performance, CUDA and OpenCL have been shown to be marginally faster than GLSL assuming efficient memory access (Amorim et al., 2009; Vassilev, 2010). Due to interoperability and only minimal performance loss, GLSL was the chosen GPGPU programming interface for our implementation.

In order to use GLSL for GPGPU computation versus traditional computer graphics applications, certain techniques need to be leveraged: offscreen rendering, direct memory access, and multipass rendering. Offscreen rendering allows OpenGL scenes to be rendered into buffers other than the standard frame buffer or screen. This is done by creating a frame buffer object (FBO) and binding its output to the desired output buffer, such as an OpenGL texture. When geometry is rendered through the pipeline, it will output into the texture versus the screen. By rendering a screen aligned quad with the FBO bound, a GLSL fragment program can be run for every pixel in the output buffer, allowing per pixel computation (Fung and Mann, 2004).

In order to get input into the GLSL program, buffers such as textures are bound that the program can access. When using GPUs, one of the major bottlenecks are transfers of data to and from the GPU buffers. To alleviate this bottleneck, we use direct memory access (DMA), which allows specifically allocated parts of memory to be used in transfers (Shreiner and Group, 2009). Transfers through DMA do not require the CPU and GPU to synchronize, and the GPU can transfer data while simultaneously process its pipeline. Thus, utilizing a DMA approach mitigates the bottleneck of transfers.

Lastly, multipass rendering is leveraged to run different GLSL programs multiple times on different buffers, achieving synchronization of threads between multiple stages in a pipeline. By clearing or not using depth buffering, the OpenGL driver will not cull any geometry from the processing pipeline. Transforms on the data can be utilized by binding different input and output buffers as well as different GLSL programs in between rendering screen aligned quads. This allows previously computed data to be utilized in future stages, as well as compounding effects and is known as multipass rendering since multiple rendering passes are used to render a single scene.

4.4 Implementation

In the 3D decoding pipeline for our system, there are seven discrete stages: phase wrapping, phase filtering, phase unwrapping, phase filtering, depth map calculation, normal map calculation, and final rendering. Figure 4.2 illustrates the decoding pipeline, showing the data at each step. This section will look into the implementation of each step.

4.4.1 Step 1: Phase Wrapping

Phase wrapping is the first step in the overall pipeline that takes incoming fringe data from the camera and wraps it into wrapped phase maps. Each set of three step fringe images are passed in as a texture, with the three images in the red, green, and blue color channel. Next, Equation (4.4) is applied to each image resulting in two wrapped phase

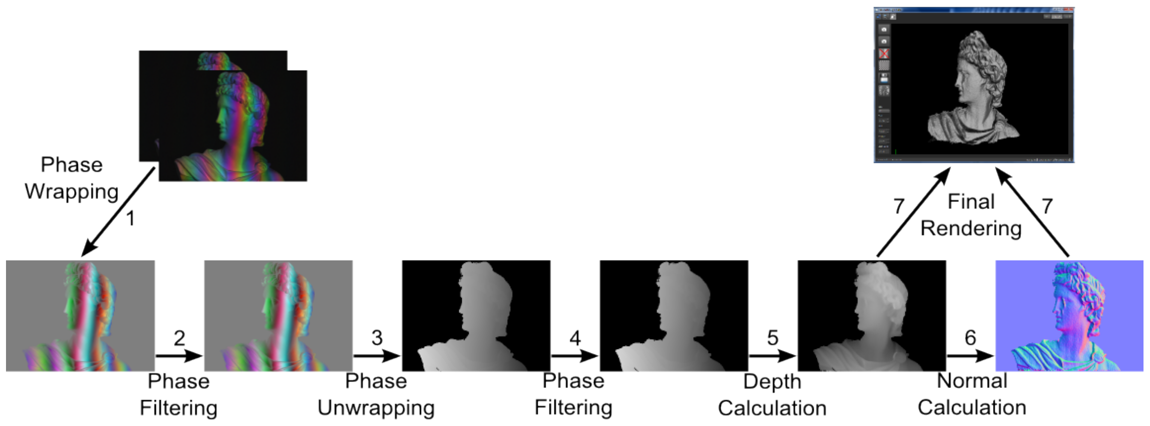


Figure 4.2 Real-time 3D scanning pipeline. The pipeline starts with the fringe images (packed into the RGB color channels of the fringe images presented) and streams them to the GPU via DMA transfers. Next phase wrapping is performed, followed by Gaussian phase filtering, phase unwrapping, median phase filtering, depth calculation, and normal calculation. Finally, final rendering is performed using the depth map and normal map producing the 3D scan.

maps, ϕ_1 and ϕ_2, one for each frequency. At this point, the sine and cosine components of the phases are taken out, and each component is rendered out to the output texture color channels (r, g, b, a) as

$$r = \sin(\phi_1), \tag{4.9}$$

$$g = \cos(\phi_1), \tag{4.10}$$

$$b = \sin(\phi_2), \tag{4.11}$$

$$a = \cos(\phi_2). \tag{4.12}$$

The components are extracted so that during phase filtering, errors are not introduced into the wrapped phase map.

4.4.2 Step 2: Gaussian Phase Filtering

Phase filtering is the only stage of the pipeline that can have a variable number of steps since it depends on how the unwrapped phase should be filtered. In our experimental pipeline, we performed one pass of a separable 11 × 11 Gaussian filter; this is done to smooth out high frequency phase noise. If larger kernels are needed multiple passes of a small kernel can be utilized, but in our analysis, it was faster to perform just a single 11×11 kernel. The separable Gaussian filter requires two rendering passes, one for the vertical pass and one for the horizontal pass of the Gaussian kernel. By using a separable Gaussian filter, only 22 texture lookups are required, 11 for horizontal and 11 for vertical. If a non separable kernel was used, 121 texture lookups would be required, substantially slowing down filtering.

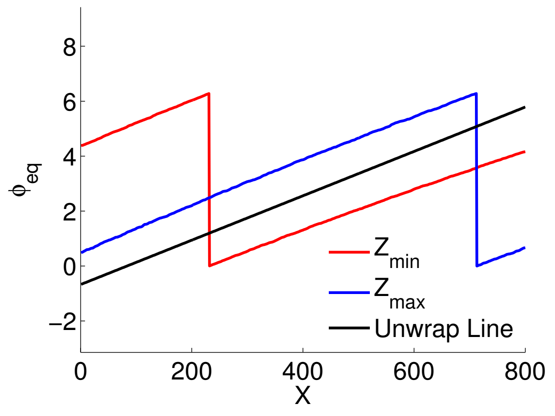
4.4.3 Step 3: Phase Unwrapping

The phase unwrapping stage involves taking the filtered wrapped phase components and combining them into an unwrapped phase map. To recover ϕ_1 and ϕ_2 the $\tan^{-1}$ is applied to the filtered sine and cosine components.

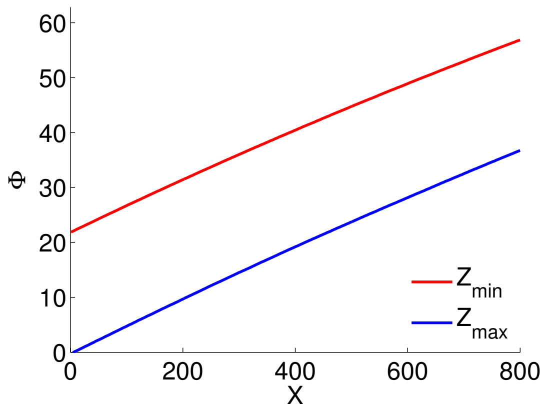
$$\phi = \tan^{-1}\left[\frac{\sin\phi}{\cos\phi}\right]. \tag{4.13}$$

At this point the equivalent phase ϕ_{eq} can be calculated using Eq. (4.5). If the fringe periods T_1 and T_2 are properly chosen, such that the equivalent fringe period T_{eq} is large enough that the single fringe spans the entire image, ϕ_1 can be unwrapped with Eq. (4.8). Choosing these optimal fringe periods is typically not easy (Towers et al., 2003a), since if these two fringe periods are too close, the equivalent phase map will be very noisy, making it difficult to resolve the phase steps (Creath, 1987).

The two-frequency phase-shifting algorithm works well for high-end systems where the sensor noise is small. To mitigate the noise induced phase unwrapping problems for our low-end system, we chose fringe periods that result in the equivalent phase only spanning half the image (i.e., it is approximately 50% less sensitive to noise effects). This selection, of course, will introduce a 2π phase jump. However, unlike the convention phase map where there are many 2π jumps per line perpendicular to the fringe direction, the proposed method only has a single phase jump at its maximum. Therefore, it is not



(a)



(b)

Figure 4.3 Cross section of captured phase before and after phase unwrapping. (a) cross section of wrapped phase ϕ_{eq}, red denotes the phase at Z_{min}, blue the phase at Z_{max}, and the black line is the phase unwrapping line; (b) cross section of unwrapped phase after unwrapping ϕeq with the phase unwrapping line, and using Φ_{eq} to unwrap ϕ_1.

necessary to adopt a complex spatial phase unwrapping algorithm to unwrap the phase map. Instead, to correct for the phase jump, we employ a parallel phase unwrapping method that unwraps the phase based on a phase value and its location. Specifically, we start with capturing the phase maps of a flat plane at two extreme depth locations, Z_{min} and Z_{max} the minimum and maximum depth values respectively, and then plotting a cross section yielding Fig. 4.3(a). As can be seen, there is a gap between the min and max phase jump that can be best separated by a line. The equation of this line is

$$y = \frac{2\pi}{P} \times x + b_{\phi}, \tag{4.14}$$

where P is the number of camera pixels per period of the fringe, and b_{ϕ} is the y intercept found though fitting the line between the phase of Z_{min} and Z_{max}. Using this equation, phase values below this line should have 2π added to them to correct for the jump, and phase values above do not. The resulting unwrapping is shown with Fig. 4.3(b). By adopting this proposed phase unwrapping algorithm, fringe periods with a larger difference can be selected, reducing the overall noise.

4.4.4 Step 4: Median Phase Filtering

Again, similar to the Step 2 (phase filtering) stage in the pipeline, this stage can have a variable number of steps since it depends on how many passes of the filter are needed for the implementation. In our implementation, we performed a single pass of a specialized median filter (McGuire, 2008), that removes one or two pixels spiking noise due to incorrect phase unwrapping that could be caused by motion or system noise. In this research, we adopted the median filter with a size of 1×5 that operated in the direction of the phase gradient, reducing the number of comparisons and branches required. Once the median for a pixel is calculated, the delta between the median phase ϕ_m and actual phase ϕ_a is taken and rounded after dividing by 2π. This results in an integer number k that can be determined by

$$k = round\left[\frac{\phi_a - \phi_m}{2\pi}\right]. \tag{4.15}$$

The none zero k indicates a spiking point that can be corrected by subtracting the k number of 2π. Our research found that this filtering stage effectively remove spiking noise yet will not introduce artifacts caused by standard median filtering.

4.4.5 Step 5: Depth Calculation

Depth map calculation involves calculating depth values for each unwrapped phase value. There are a number of approaches (Zhang and Huang, 2006b; Xiao et al., 2012; Villa et al., 2012; Wen et al., 2010; Legarda-Sáenz et al., 2004; Cuevas et al., 2000; Li et al., 2008; Gao et al., 2008; Yang et al., 2008; Hu et al., 2003; Huang et al., 2010) to do this based on the chosen calibration, and in our method we chose to perform a very simple reference-plane-based approach detailed by Xu et al. (Xu et al., 2011). By capturing the phase of a reference plane Φ_R where $z = 0$, a phase difference between the captured phase Φ_C and Φ_R can be calculated. This phase difference will be proportional to the depth z by a scaling value To calculate this in the fragment shader, a texture containing

Φ_R is read in along with a texture containing the filtered phase Φ_C. Subtracting the two phases and scaling, based on a scaling factor c determined through calibration, yields the depth value z; this depth value is then rendered out, yielding the depth map.

4.4.6 Step 6: Normal Map Calculation

During the reconstruction, point normals for the geometry are calculated so that Phong lighting may be applied. The normal map is calculated by calculating all adjacent surface normals and then averaging them together, resulting in a point normal. Adjacent surface normals are calculated by taking the vectors between the current coordinate and two neighboring coordinates, moving sequentially counterclockwise in a 3 $\times$ 3 neighborhood, and calculating the cross product. This yields a surface normal for the polygon comprised of these three points. After normalizing and averaging all these surface normals, the result is the point normal for the coordinate. This is rendered out to the normal map texture, yielding a normal map for the scanned data.

4.4.7 Step 7: Final Rendering

The last stage in the 3D scanning pipeline is final rendering. Before final rendering can take place, the frame buffer needs to be switched back from the FBO to the screen so that the result is rendered to the screen. After doing so, the depth map and normal map are passed to the final render shader and a plane of points is rendered with uniformly varying texture coordinates. At this point, the final geometry can be down-sampled, if needed, by rendering a reduced number of points in the plane of points. In the vertex shader the depth for the point is read from the depth map and the vertex z attribute is modified accordingly. In the fragment shader the point normal is read from the normal map, and then per fragment Phong shading is applied to each point. At this point the reconstructed geometry is rendered onto the screen.

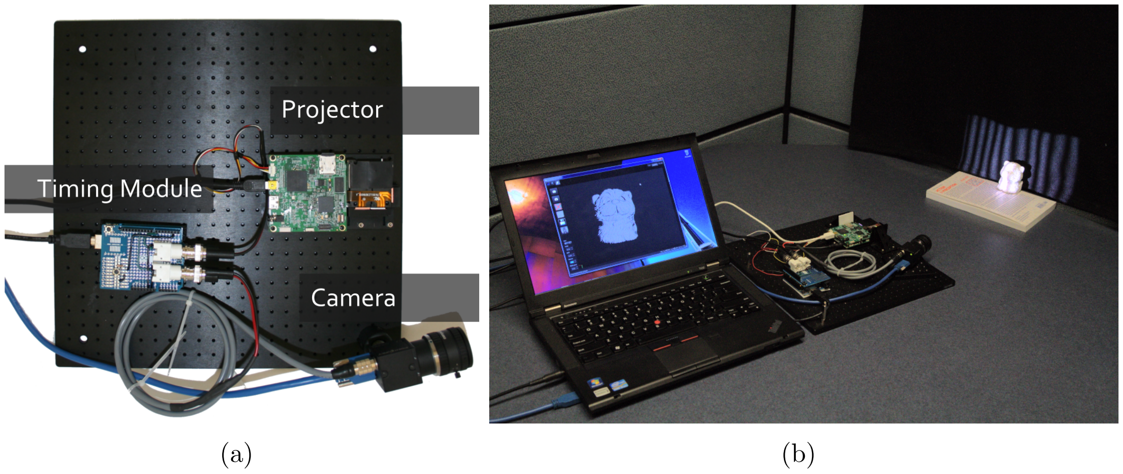


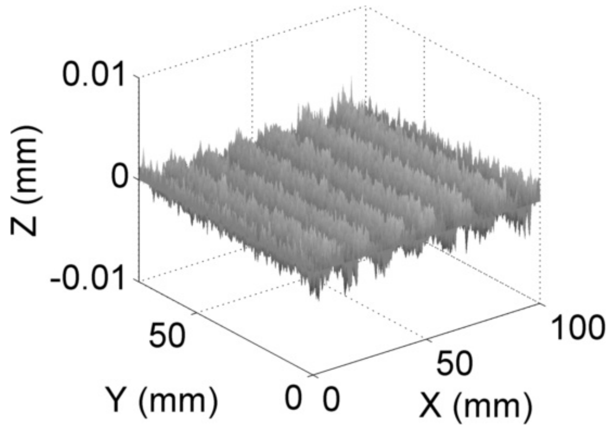
Figure 4.4 Picture of the developed system. (a) Overview of the system with labeled components; (b) System scanning static sculpture.

4.5 Experimental Results and Discussion

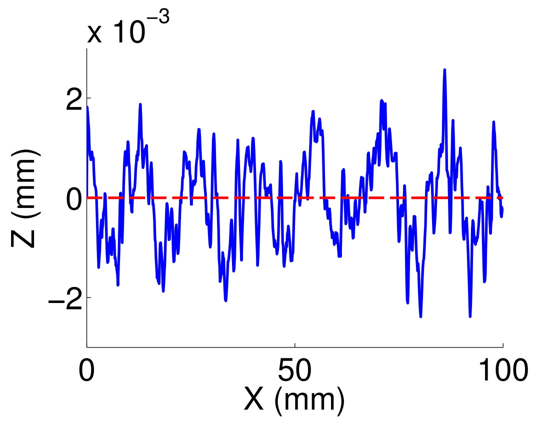
To test the effectiveness of the system we performed different experiments including, measuring a flat surface, measuring static sculptures, and measuring dynamically moving objects. In each of the experiments the hardware stayed consistent, a Point Grey Research Flea3 camera with a Computar 12 mm lens, a Texas Instruments digital light processing (DLP) LightCrafter projector, an Arduino Uno for timing signal generation, and a IBM Lenovo laptop with a Intel i5 3320M 2.6GHz CPU and NVIDIA Quadro NVS5400M GPU. The DLP Lightcrafter can project and switch binary structured patterns at 4 kHz with full resolution, and the Point Grey camera can acquire images up to 180 Hz with an image resolution of 800 $\times$ 600 and exposure time of less than 2 ms operating under the external triggering mode. Figure 4.4 shows an overview of the system with labeled components as well as the system scanning a sculpture.

Since under the external triggering mode, the maximum exposure time of the camera can use is 2 ms at 180 Hz capturing rate, the conventional sinusoidal fringe projection technique does work due to the rigorous timing requirement (i.e., the camera exposure time must be precisely 1000/180 ms). Therefore, the binary defocusing technique (Lei and Zhang, 2009) was used to circumvent this problem. Furthermore, to alleviate the short depth range problem caused by square binary defocusing technique, we adopted the error-diffusion dithering method to generate all six desired sinusoidal fringe patterns (Wang and Zhang, 2011); and the modified two-frequency phase-shifting algorithm to unwrap the phase pixel by pixel. For such a system, since it requires six fringe images to recover one 3D shape, the 3D shape measurement rate is 30 fps.

To test the system noise, we first captured a flat surface. In an ideal system, the surface should be perfectly flat, but due to sensor and environment noise their are small variations. Figure 4.5(a) shows the results of the capture, and Fig. 4.5(b) shows a horizontal cross section at the 300-th row. The variations in the surface height results in a root-mean-square (RMS) error of 0.00081 mm, for a measurement area of 100 mm × 75 mm with a resolution of 800 × 600.



(a)



(b)

Figure 4.5 Measurement result of a flat surface. The measured area is approximately 100 mm × 75 mm, and the resulting rms error is 0.00081 mm. (a) 3D plot of the surface; (B) example cross section of the surface.

To test the system's capabilities of measuring more complex 3D objects, we performed measurements on a static sculpture. Figure 4.6(a) shows the statue we captured, and the surface geometry is pretty complex. The live reconstructed 3D results on the computer screen are shown in Figs. 4.6(b) and 4.6(c). These results clearly show there are no phase jumps, verifying that the proposed phase unwrapping algorithm works properly. These images are also very clean without spiking noise, common to a multi-frequency phase-shifting algorithm, meaning that the proposed filtering methods can effectively remove spiking noise.

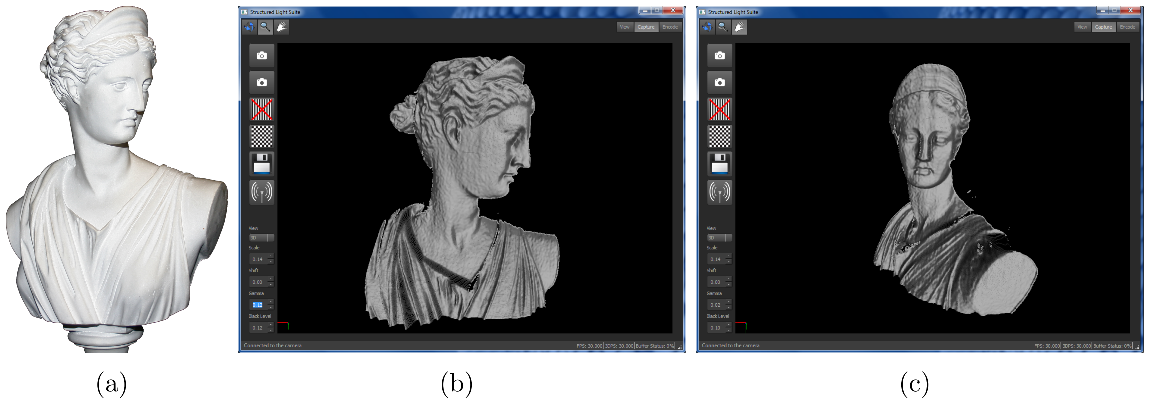


Figure 4.6 Capture of static statues. (a) 2D photo of statue; (b)-(c) Two screen shots of the 3D reconstructions of the statue.

To further demonstrate the speed of the proposed system. We measured some dynamically changing objects. Figure 4.7 and the associated medias show two examples, one single hand motion capture, and simultaneous two-hand motion capture. The videos were filmed from the computer screen using an HD video recorder (Sony: HDR-AX2000) so as not to affect the reconstruction frame rate. The computer screen was the Lenovo laptop screen that demonstrates the live reconstructed 3D results of the object being measured. Regardless of the geometry complexity, this laptop can constantly reconstruct and display 3D geometry at 30 fps with an image resolution of 800 $\times$ 600. These experiments once again confirm that the proposed GPU phase-shifting algorithm, and portable system can deliver high-resolution, real-time 3D shape measurement of dynamically deformable objects with arbitrary shapes.

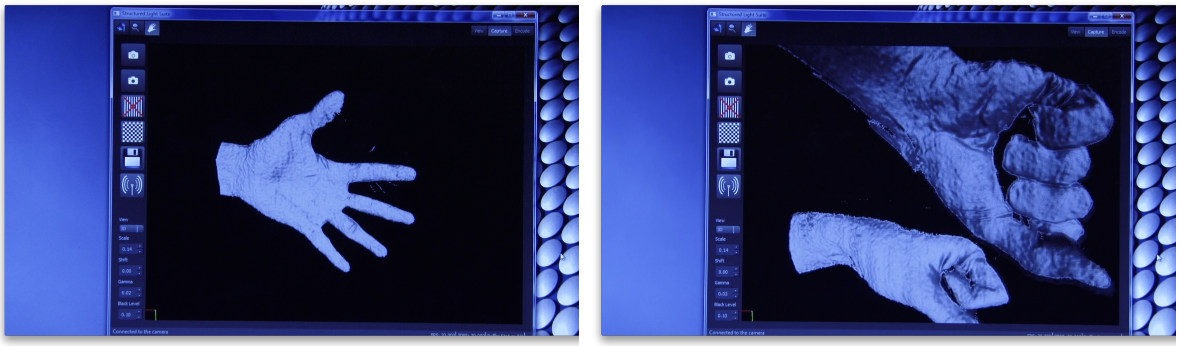

(a) (b)

Figure 4.7 Example frames from capturing dynamically moving objects. (a) A single hand motion (Media 1); (b) Two hands motion (Media 2)

4.6 Conclusion

This paper has presented a technique for achieving high-resolution, real-time 3D shape measurement on a portable device by implementing the entire processing pipeline of a modified two-frequency phase-shifting algorithm on a GPU. We have demonstrated the principles behind the techniques leveraged by the system, as well as giving a description of the GPU implementation. By utilizing a GPU for the entire 3D processing and display process, the processing power requirements of CPU have been drastically reduced, allowing the system to be realized with a portable device. Through experiments, we have shown that 3D shape acquisition, reconstruction, and display can reach 30 fps on a Lenovo laptop at an image resolution of 800 $\times$ 600. Due to utilization of the binary defocusing technique, the USB 3.0 camera, and the GPU implementation, the whole system is quite inexpensive, making such a system potentially accessible to the general public.

Acknowledgments

This material is based upon work supported by the National Science Foundation (NSF) Graduate Research Fellowship under Grant No. DGE1247194, and the NSF under grant numbers CMMI-1150711 and CMMI-1300376. Any opinion, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the NSF.

CHAPTER 5. Composite phase-shifting algorithm for 3-D shape compression

A paper published in the Journal of Optical Engineering in June 2010.[1]

Nikolaus Karpinsky[2,4] and Song Zhang[3,4]

5.1 Abstract

With recent advancements in 3-D imaging and computational technologies, acquiring 3-D data is unprecedentedly simple. However, the use of 3-D data is still limited due to the size of 3-D data, especially 3-D video data. Therefore, the study of how to store and transmit the 3-D data in real time is vital. In this research, we address a technique that encodes a 3-D surface shape into a single 24-bit color image. In particular, this image is generated by advanced computer graphics tools with two primary color channels encoded as sine and cosine fringe images, and the third channel encoded as a stair image to unwrap the phase obtained from the two fringe images. An arbitrary 3-D shape can then be recovered from a single image. 3-D shapes with differing levels of complexity have been tested along with various image formats. Experiments demonstrated that, without significantly losing the shape quality, the compression ratio can go up to 36.86:1.

[1]This paper was published in the Journal of Optical Engineering and is made available as an electronic reprint with the permission of SPIE. The paper can be found at the following URL on the SPIE website: http://opticalengineering.spiedigitallibrary.org/article.aspx?articleid=1096370. Systematic or multiple reproduction of distribution to multiple locations via electronic or other means is prohibited and is subject to penalties under law.

[2]Primary researcher and author

[3]Assistant professor and advisor.

[4]Department of Mechanical Engineering, Iowa State University

5.2 Introduction

With recent advancements in 3-D imaging and computational technologies, acquiring 3-D data is unprecedentedly simple. During the past few years, advancements in digital display technology and computers has accelerated research in 3-D imaging techniques. The 3-D imaging technology has been increasingly used in both scientific studies and industrial practices. Real-time 3-D imaging recently emerged, and a number of techniques have been developed (Rusinkiewicz et al., 2002; Guan et al., 2003; Zhang et al., 2003; Davis et al., 2003; Zhang and Huang, 2006a). For example, we have developed a system to measure absolute 3-D shapes at 60 frames/sec with a image resolution of 640 $\times$ 480 (Zhang and Yau, 2007b). The 3-D data throughput of this system is approximately 228 MB/sec, which is very difficult to store and transmit simultaneously. Therefore, research in how to store and transmit the 3-D data in real time is vital.

Unlike 2-D images, 3-D geometry conveys much more information, albeit at the price of increased data size. In general, for a 2-D color image, 24 bits (or 3 bytes) are enough to represent each color pixel (red (R), green (G), and blue(B)). However, for 3-D geometry, an (x, y, z) coordinate typically needs at least 12 bytes excluding the connectivity information. Thus, the size of 3-D geometry is at least 4 times larger than that of a 2-D image with the same number of points. However, since conventional general 3-D shape formats (including OBJ, STL) require connectivity information, 3-D shape data is usually on the order of 10-20 times larger than a 2-D image with the same number of points. Table 5.1 gives a comparison of various 3-D shape formats. Comparing the formats, MAT5 gives the best compression as it was designed specifically to store point cloud data from 3-D scanners disregarding polygon links (Hassebrook, 2010); even this format is over 25 times larger than a Holoimage.

Another benefit of the Holoimage format is that it can use existing research of 2-D image processing. 2-D image processing is well studied field, and the size of 2-D images

	Holoimage	MAT5	PLY	DAE	OBJ	STL
File size:	210KB	5.5MB	6.5MB	10.6MB	12.8MB	17MB
Ratio:	1:1	1:26.19	1:30.95	1:50.48	1:60.95	1:80.95

Table 5.1 Compression comparison of various 3-D formats compared to the Holoimage format. Formats contain only vertices and connectivity if required, and are in binary format if applicable to the format; no point normals or texture coordinates are stored.

is much smaller than that of 3-D geometries. The idea of a reduced data size and existing techniques for 3-D image process is attractive. Since 3-D geometry is usually obtained by 2-D devices (e.g., a digital camera), it is natural to use its originally acquired 2-D format to compress it.

In this research, we address a technique that converts 3-D surfaces into a single 2-D color image. The color image is generated using advanced computer graphics tools to synthesize a digital fringe projection and phase-shifting system for 3-D shape measurement. We propose a new coding method named "composite phase-shifting algorithm" for 3-D shape recovery. With this method, two color channels (R, G) are encoded as sine and cosine fringe images, and the third color channel (B) is encoded as a stair image; the stair image can be used to unwrap the phase map obtained from two fringe images point by point. By using a 24-bit image and no spatial phase unwrapping, the 3-D shape can be recovered; therefore the single 2-D image can represent a 3-D surface.

The encoded 24-bit images can be stored in different formats, e.g., bitmap, portable network graphics (PNG), and JPG. If the image is stored in a lossless format, such as bitmap and PNG, the quality of 3-D shape is not affected at all. We found that lossy compression such as JPG compression cannot be directly implemented, as it distorts the blue channel severally affecting the 3-D surface. To circumvent this problem, red and green channels are stored using JPG under different compression levels while the blue channel remains in a lossless PNG format. Our experiments demonstrated that there is little error for a compression ratio up to 36.86:1. Experiments will be presented to verify the performance of the proposed approach.

Section 5.3 presents the fundamentals of the virtual fringe projection system and the composite phase-shifting algorithm. Section 5.4 shows experimental results, and finally Section 5.5 summarizes the paper.

5.3 Principle

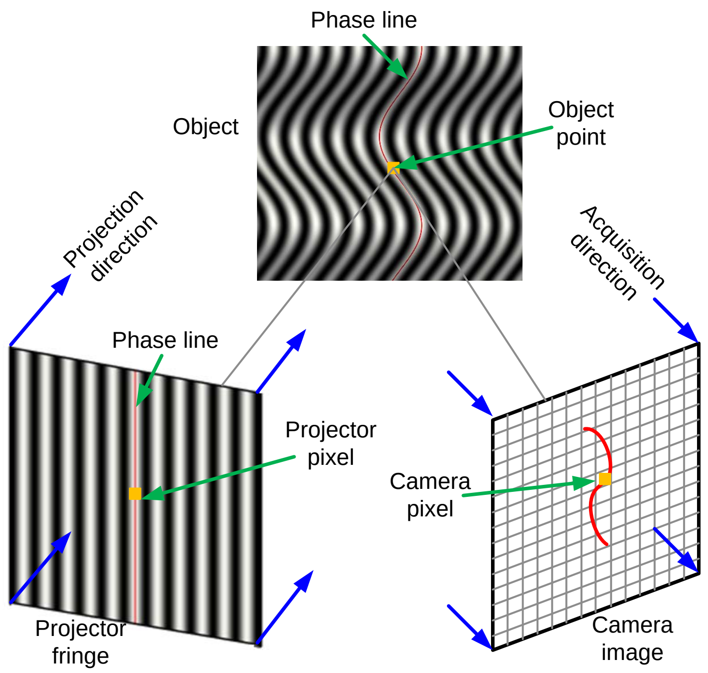


Figure 5.1 Virtual digital fringe projection system setup.

5.3.1 Virtual digital fringe projection system setup

Figure 5.1 shows a the virtual fringe projection system setup, which is also known as a Holoimage system (Gu et al., 2006). It is very similar to that a real fringe projection based 3-D shape measurement system. A projector projects fringe images onto an object and a camera captures the fringe images that the object has distorted. 3-D information can be retrieved if the geometric relationship between the projector pixels and the camera pixels are known.

The virtual system differs from the real 3-D shape measurement system in that the projector and the camera are orthogonal devices instead of perspective ones, and the relationship between the projector and the camera are precisely defined. Thus, the shape reconstruction becomes significantly simplified and precise. To represent an arbitrary 3-D shape, a multiple-wavelength phase-shifting algorithm (Towers et al., 2003b, 2005; Cheng and Wyant, 1985; Upputuri et al., 2009) can be used. However, it requires more than three fringe images to represent one 3-D shape, which is not desirable for data compression.

5.3.2 Composite phase-shifting algorithm

Due to the virtual nature of the system, all environmental variables can be precisely controlled, simplifying the phase-shifting process. To obtain phase, only sine and cosine images are actually needed, which can be encoded into two color channels, e.g., red, green channels. The intensity of these two images can be written as,

$$I_r(x,y) = 255/2[1+\sin(\phi(x,y))], \tag{5.1}$$

$$I_g(x,y) = 255/2[1+\cos(\phi(x,y))]. \tag{5.2}$$

From the previous two equations, we can obtain the phase

$$\phi(x,y) = \tan^{-1}\left[\frac{I_r - 255/2}{I_g - 255/2}\right]. \tag{5.3}$$

The phase obtained in Eq.(5.3) ranges $[-\pi, +\pi)$. To obtain a continuous phase map, a conventional spatial phase unwrapping algorithm can be used. However, it is known that the step height changes between two pixels cannot be larger than π. The phase unwrapping step is essentially to find the integer number (K) 2π jumps for each pixel so that the true phase can be found (Ghiglia and Pritt, 1998)

$$\Phi(x,y) = 2\pi K + \phi(x,y). \tag{5.4}$$

If an additional stair image, $I_b(x, y)$, is used whose intensity changes are precisely aligned with the 2π phase jumps (as shown in Fig. 5.2), the phase unwrapping step can be performed point by point by using the stair image information. In other words, the unwrapped phase will be

$$\Phi(x, y) = 2\pi I_b(x, y) + \phi(x, y). \tag{5.5}$$

In practice, to reduce the problems caused by digital effects, instead of using one grayscale value for each increment, a larger value is used. In the example shown in Fig. 5.2, 80 grayscale values are used to represent one stair.

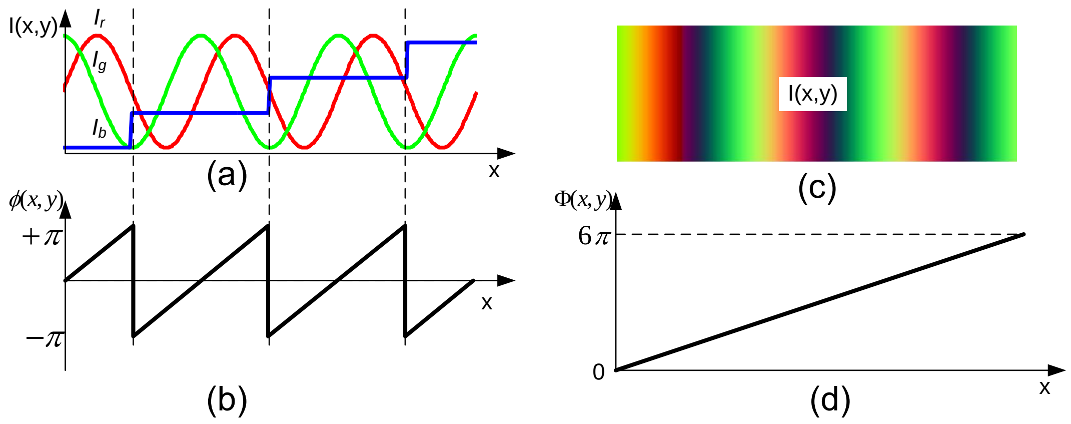


Figure 5.2 Schematic diagram of the proposed composite algorithm for single color fringe image generation. (a) Cross section of the color fringe images, red is the sine image (I_r), green is the cosine image (I_g) and blue is the stair image (I_b); (b) The cross section of the phase map ($\phi(x, y)$) using red and green fringe images from Eq. 5.3; (c) The real fringe image; (d) The unwrapped phase after correcting the wrapped phase ($\phi(x, y)$) by the stair image (I_b).

5.3.3 Phase-to-coordinate conversion

Figure 5.3 illustrates the phase-to-coordinate conversion. To explain the concepts, a reference plane (a flat surface with $z = 0$) is used. Assume the fringe pitch generated by the projector is P and the projection angle is θ, the fringe pitch on the reference plane will be $P_r = P/\cos\theta$. For an arbitrary image point K, if there is no object in place, the

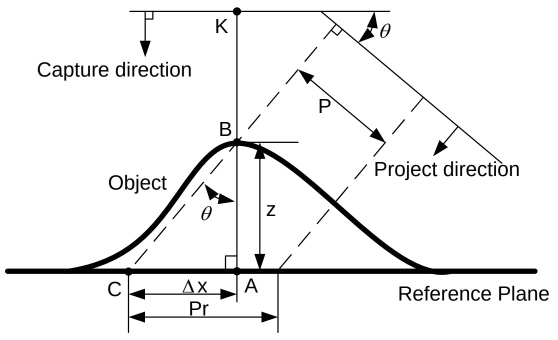


Figure 5.3 Schematic diagram for phase to coordinate conversion.

phase is Φ^r_A on the reference plane. Once the object is in position, the imaging point on the object is B. From the projector point of view, B on the object and C on the reference plane have the same phase, i.e., $\Phi = \Phi_B = \Phi^r_C$. Then, we have

$$\Delta\Phi = \Phi^r_C - \Phi_A = \Phi^r_B - \Phi^r_A = \Phi - \Phi^r_A. \tag{5.6}$$

Since the fringe stripes are uniformly distributed on the reference plane. For the pipeline introduced in this paper, the reference plane is well defined ($z = 0$). The phase on the reference plane is defined as a function of the projection angle θ and the fringe pitch P,

$$\Phi^r = 2\pi i/P_r = 2\pi i \cos\theta/P, \tag{5.7}$$

assuming phase 0 is defined at $i = 0$ and the fringe stripes are vertical. Here, i is the image index horizontally. From Eqs.(5.6) and (5.7), we have,

$$\Delta\Phi = \Phi - 2\pi i \cos\theta/P. \tag{5.8}$$

Also we have,

$$\Delta\Phi = \Phi^r_C - \Phi^r_A = 2\pi\Delta i \cos\theta/P. \tag{5.9}$$

Moreover, the graphics pipeline can be configured to visualize within a unit cube, when pixel size is $1/W$. Here, W is the total number of pixel horizontally, or window width. Then

$$x = i/W, \tag{5.10}$$

assume the origin of the coordinate system is aligned with the origin of the image.

Similarly, for the y coordinate, assume y direction has the same scaling factor, we have

$$y = j/W, \tag{5.11}$$

here j is the image index vertically.

From the geometric relation of the diagram in Fig. 5.3, it is obvious that

$$z = \Delta x / \tan\theta. \tag{5.12}$$

Combining this equation with Eqs (5.9) and (5.10), we will have

$$z = \frac{P\Delta\Phi}{2\pi W \sin\theta}. \tag{5.13}$$

Finally, the equation governing the z coordinate calculation is

$$z = \frac{P(\Phi - 2\pi i \cos\theta / P)}{2\pi W \sin\theta}, \tag{5.14}$$

which is a function of the projection angle θ, the fringe pitch P, and the phase Φ obtained from the fringe images.

5.4 Experiment

To verify the performance of the proposed approach, we first tested a sphere with a diameter of 1 mm (unit can be any since it is normalized into a unit cube) as shown in Fig. 5.5, whose color fringe image is shown as Fig. 5.4(a). In this example, we used a stair step height of 5, projection angle of $\theta = 30°$, and a fringe pitch of $P = 16$ pixels. All of the fringe images used in the rest of the paper, have exactly the same setup. From the red and green channels, the phase map can be calculated by Eq. (5.3), which is shown in Fig. 5.4(b). The blue channel (shown in Fig. 5.4(c)) is then applied to unwrap the phase map point by point using Eq. (5.5), the result is shown in Fig. 5.4(d). On this unwrapped phase map, there are some artifacts (white dots) that are not clearly shown in this figure (will be more clearly shown in Fig. 5.4(e)). Using the phase-to-coordinate conversion algorithm introduced in Subsection 5.3.3, the phase map can be converted to 3-D, which is shown in Fig. 5.4(e). The artifacts (spikes) are more obvious. It is caused by the sampling of the projector and the camera. Because the projector and the camera are digital devices, the discrete signal of the fringe images and the stair image introduce a subpixel shift between the jumps. Fortunately, because this shift is limited to 1 pixel either left or right, this problem can be fixed using a conventional image processing technique, e.g., median filtering in phase domain. Figure 5.4(f) shows the corrected result.

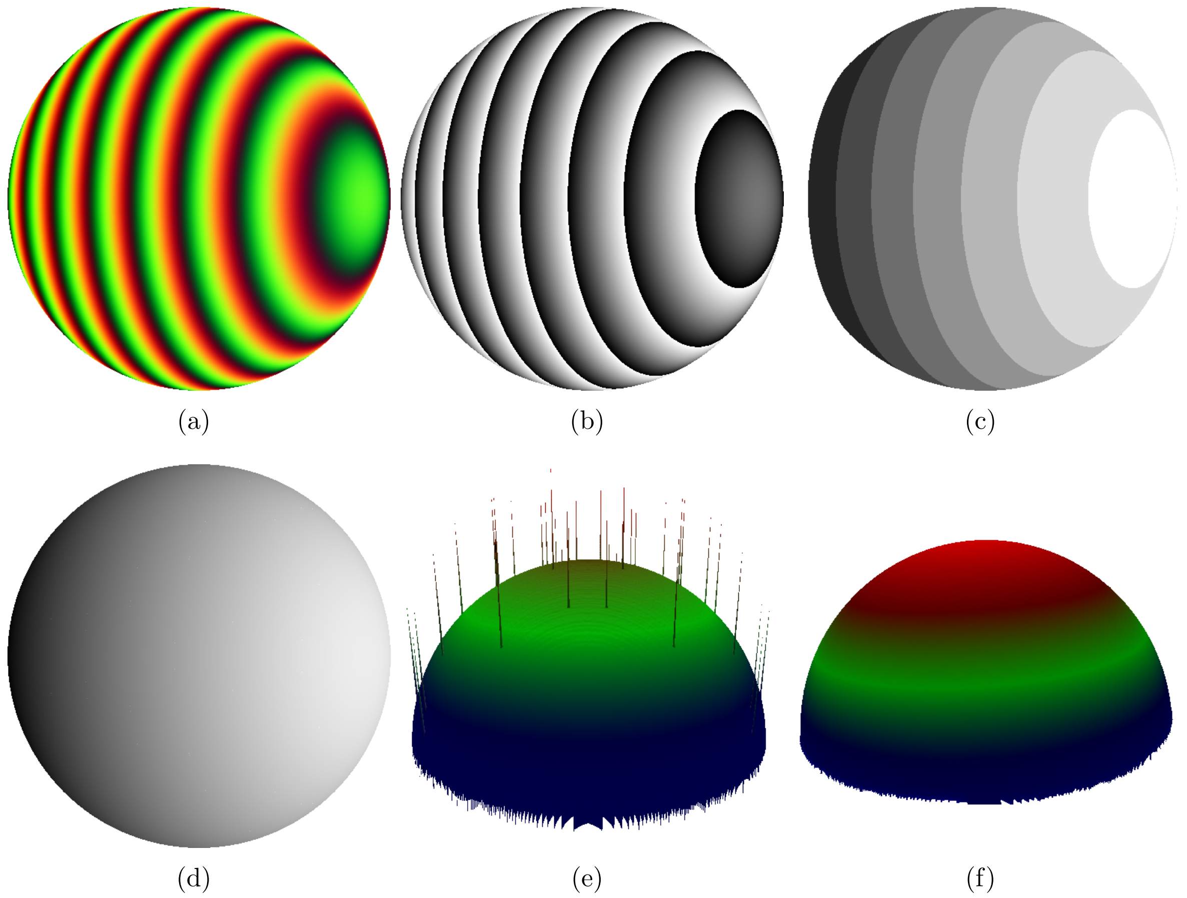


Figure 5.4 3-D recovery using the single color fringe image. (a) Fringe image; (b) Phase map using red and green channels of the color fringe image; (c) Stair images (blue channel); (d) Unwrapped absolute phase map; (e) 3-D shape before applying median filtering; (f) 3-D shape after applying median filtering.

The cross section of the reconstructed 3-D shape and the theoretical sphere is shown in Fig. 5.5(a), and the difference is shown in Fig. 5.5(b). It is very obvious that the difference is negligible.

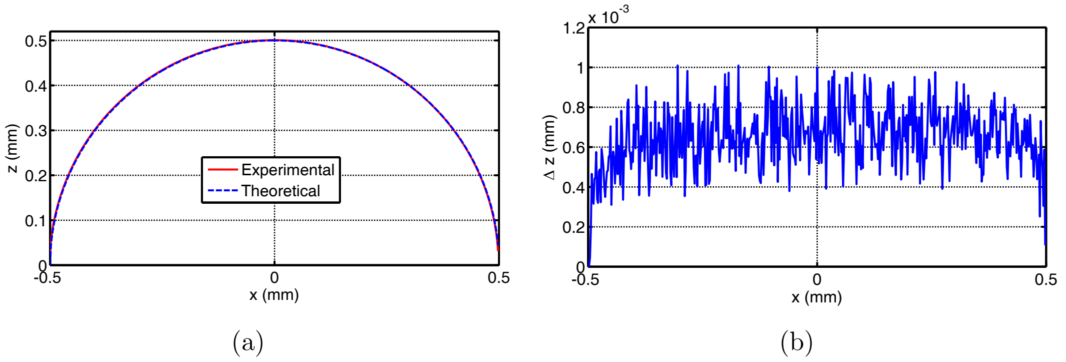


Figure 5.5 Comparison between the reconstructed 3-D shape and the theoretical one. (a) Cross section of 256th row; (b) Difference (RMS: 1.68×10^{-4} mm or 0.03%).

Because this algorithm allows point by point phase unwrapping, it can be used to reconstruct arbitrary shapes of an object with an arbitrary number of steps. To verify this, we tested a step-height surface: a flat object with a deep squared hole. The color image is shown in Fig. 5.6(a), even though the object has height variations greater than the step height, the fringe image does not appear to have discontinuities; this is because the virtual system is different from real 3-D shape measurement system in that the light can pass through objects. The phase map obtained from the fringe images is shown in Fig. 5.6(b), the phase jumps are very obvious. Because this technique uses the third channel to unwrap the phase, the 3-D shape can be correctly reconstructed, which is shown in Fig. 5.6(c). This 3-D shape has large height variations, greater than one period of phase range, yet is correctly reconstructed. Figure 5.7 shows the cross section of the 3-D shape.

An actual scanned 3-D object is then tested for the proposed algorithm. Figure 5.8 shows the experimental result. The original shape is shown in Fig. 5.8(a), the color fringe image is shown in Fig. 5.8(b), the unwrapped phase map and the recovered 3-D shape are

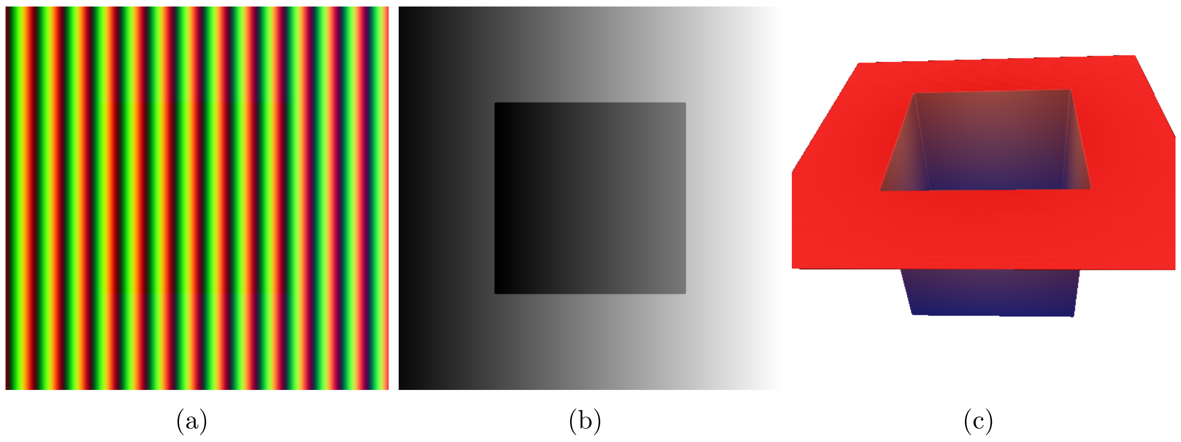


Figure 5.6 3-D recovery for step height object. (a) Color fringe image; (b) Unwrapped phase map; (c) 3-D shape.

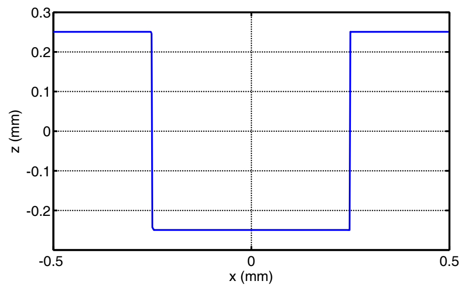


Figure 5.7 256th row of the step height object.

shown in Fig. 5.8(c) and Fig. 5.8(d), respectively. If the original shape and the recovered shape are rendered in the same window, the results are shown in Fig. 5.8(e) in shaded mode and Fig. 5.8(f) in wireframe mode. It clearly demonstrates that the recovered 3-D shape and the original shape are almost perfectly aligned, that is, the recovered 3-D shape and the original 3-D shape do not have significant difference.

All these experiments demonstrate that the proposed single image technique can be used to represent an arbitrary 3-D surface shape, thus can be used for shape compression. We performed further experiments that use different image formats and compare the 3-D reconstruction quality. In this research, we tested Bitmap, PNG, and differing compression levels of JPG. A typical 3-D surface shown in Fig. 5.8(a) is used to verify the performance. For a 512 × 512 3-D surface together, storing x, y, z coordinates and

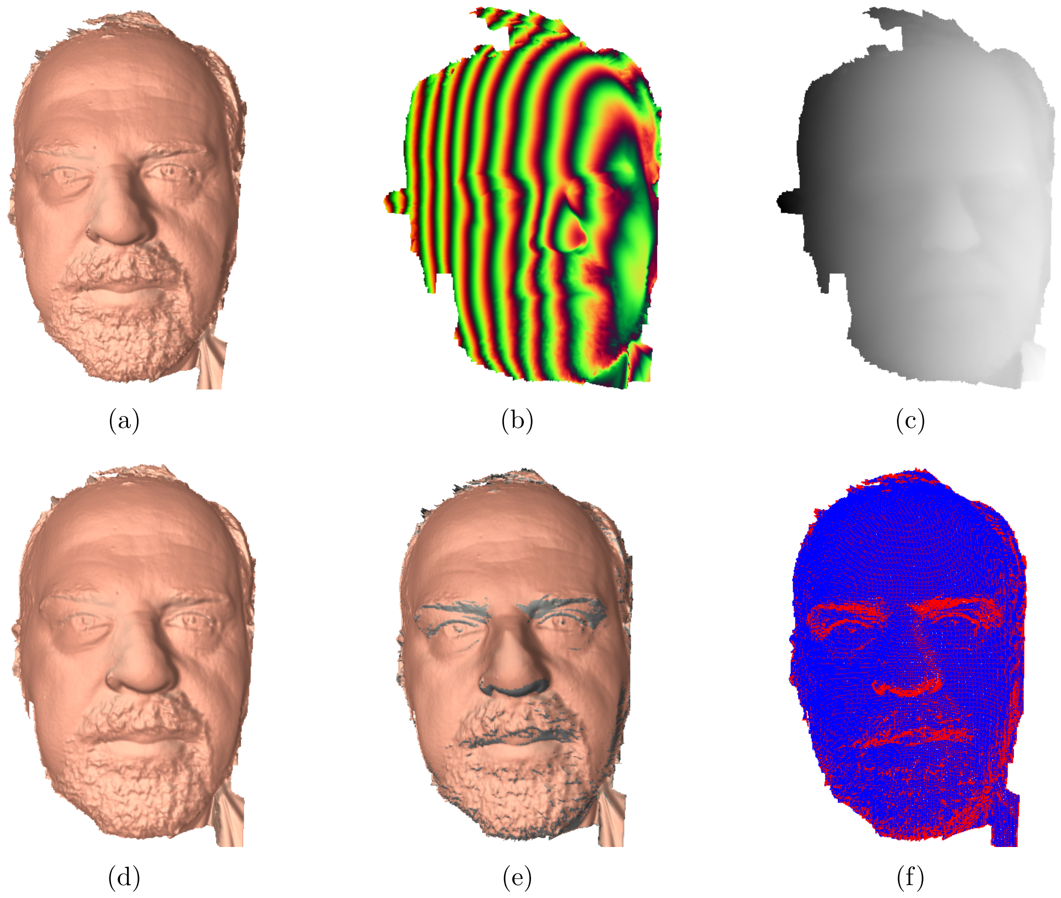


Figure 5.8 3-D recovery using the color fringe image for scanned data. (a) 3-D scanned original data; (b) Color fringe image; (c) Unwrapped phase map; (d) 3-D reconstructed shape; (e) Overlap original 3-D shape (yellow) and the recovered 3-D shape (gray) in shaded mode; (f) Overlap original 3-D shape (blue) and the recovered 3-D shape (red) in shaded mode.

the mask information will be at least 3,407,872 bytes (4 bytes floating point for each coordinate, and 1 byte for mask). Most popular 3-D formats, such as OBJ, STL, use much more space. The bitmap color image has a size of 786,486 bytes, which is approximately 4.33 times smaller. In this experiment, we use the bitmap color image as it is uncompressed lossless.

Fig. 5.9 shows the results. Portable network graphics (PNG) image format was first used to compress the image data. Since the PNG format is lossless, the original 3-D data can be recovered without any loss, while the file size is reduced to 171,257 bytes (compression ratio of 19.90:1). Fig. 5.9(a) shows the reconstructed 3-D shape and Fig. 5.9(e) shows the difference between the reconstructed 3-D shape and the original 3-D shape. It can be seen that there is no difference at all. We found that the color image cannot be directly compressed into JPG format because the third channel (blue) is intolerant of noise. To circumvent this problem, we compress red and green channels using a JPG format while retaining the blue channel in PNG format. In this manner, the file size is reduced to 92,446 bytes while retaining the 3-D shape quality with a compression ratio 36.86:1. Fig. 5.9(b) and Fig. 5.9(f) shows the reconstructed 3-D shape and the difference map, respectively. When we further compress red and green channels to a size of 92,192 byte, the image quality slightly drops, as shown in Fig. 5.9(c) and Fig. 5.9(g). It is interesting to notice that the boundary dropped more than the inside of the shape, because the boundary has sharp edges. We also demonstrated that when the file size is further reduced to 81,713 bytes, the 3-D shape quality is reduced substantially. The results are show in Fig. 5.9(d) and Fig. 5.9(h). This experiment shows that the color image can be substantially compressed without losing the data quality.

5.5 Conclusion

In this research, we successfully demonstrated that an arbitrary 3-D shape can be represented as a single color image, with red and green channel being represented as sine and cosine fringe images, and the blue channel encoded as a phase unwrapping stair function. Storing 3-D geometry in a 2-D color image format allows for conventional image compression methods to be employed to compress the 3-D geometry. However, we found that lossy compression algorithms cannot be incorporated because of the the third channel containing sharp edges. Lossless image formats, such as PNG or bitmap must be used to store the blue channel because it contains sharp edges, while red and green channels can be stored in any image format. We have demonstrated that with a

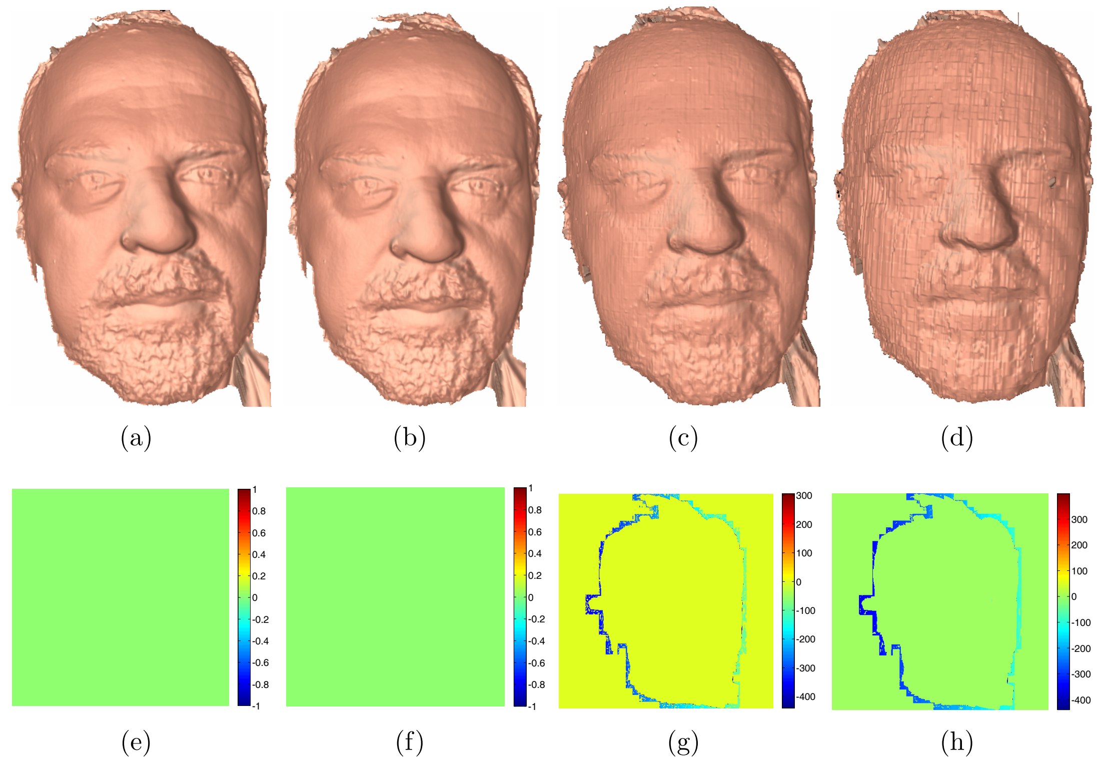


Figure 5.9 3-D reconstruction under different compression ratio. (a) PNG format (19.90:1); (b) JPG+PNG format (36.86:1); (c) JPG+PNG format (36.96:1); (d) JPG+PNG format (41.71:1).

compression ratio of 36.86:1, the shape quality did not reduce at all. In this research, by compressing 3-D geometry into 24-bit color images, the compression ratio is very high. However, after conversion, the original 3-D data connectivity information is lost and the data is re-sampled. It should be noted that because the shape reconstruction can be conducted pixel by pixel, it is very suitable for parallel processing, thus allowing for real-time shape transmission and visualization. In the future, we will explore higher image compression methods to store the red and green channels, and investigate higher compressed lossless strategies to store blue channel.

CHAPTER 6. Holovideo: Real-time 3D video encoding and decoding on GPU

A paper published in the Journal of Optics and Lasers in Engineering in February 2012.[1]

Nikolaus Karpinsky[2,4] and Song Zhang[3,4]

6.1 Abstract

We present a 3D video-encoding technique called Holovideo that is capable of encoding high-resolution 3D videos into standard 2D videos, and then decoding the 2D videos back into 3D rapidly without significant loss of quality. Due to the nature of the algorithm, 2D video compression such as JPEG encoding with QuickTime Run Length Encoding (QTRLE) can be applied with little quality loss, resulting in an effective way to store 3D video at very small file sizes. We found that under a compression ratio of 134:1, Holovideo to OBJ file format, the 3D geometry quality drops at a negligible level. Several sets of 3D videos were captured using a structured light scanner, compressed using the Holovideo codec, and then uncompressed and displayed to demonstrate the effectiveness of the codec. With the use of OpenGL Shaders (GLSL), the 3D video codec

[1]This paper was published in the Journal of Optics and Lasers in Engineering and is made available as an electronic reprint with the permission of Elsevier. The paper can be found at the following URL on the Elsevier website: http://www.sciencedirect.com/science/article/pii/S0143816611002466. Systematic or multiple reproduction of distribution to multiple locations via electronic or other means is prohibited and is subject to penalties under law.
[2]Primary researcher and author
[3]Assistant professor and advisor.
[4]Department of Mechanical Engineering, Iowa State University

can encode and decode in realtime. We demonstrated that for a video size of 512×512, the decoding speed is 28 frames per second (FPS) with a laptop computer using an embedded NVIDIA GeForce 9400m graphics processing unit (GPU). Encoding can be done with this same setup at 18 FPS, making this technology suitable for applications such as interactive 3D video games and 3D video conferencing.

6.2 Introduction

Advancements in 3D imaging and computational technology have made the acquisition and display of 3D data simpler. Techniques such as structured light, stereovision, and LIDAR have led the way in 3D data acquisition (Gorthi and Rastogi, 2010). In recent years, real-time 3D video scanning techniques have been increasingly more feasible (Rusinkiewicz et al., 2002; Zhang et al., 2004; Davis et al., 2003; Zhang and Huang, 2006a; Zhang et al., 2006; Li et al., 2010; Liu et al., 2010; Zhang et al., 2010; Wang and Zhang, 2011). Stereoscopic displays have made the display of 3D data a reality. However, as these fields and techniques evolve, a growing problem is being encountered: how can 3D video data be efficiently stored and transmitted?

Storage and transmission of high-resolution 3D video data has become a large problem due to the file sizes associated with 3D geometry. Standard 3D file storage techniques (e.g., STL, OBJ, PLY) do not lend themselves nicely to highly detailed, high framerate scenes. Instead, traditional 3D file storage techniques aim to store models and then animated models based on constraints of a few points, typically skeletal points Forstmann et al. (2007). This does not hold true for 3D scenes captured from 3D scanners as they consist of a large array of 3D points, all of which are animated; this animation is inherently unstructured and unconstrained making typical 3D file storage difficult. Introducing 3D models into 3D scenes only further exacerbates the problem, as both modalities need to be accounted for. How then can such scenes with both types of 3D data be encoded in a unifying way, which provides not only an efficient storage and transmission medium, but also a quick encoding and decoding of high definition (HD) data?

With the advent of structured light scanners comes the possibility to realize real-time 3D video. This makes applications such as 3D video conferencing and 3D presence a possibility. For example, the University of South California (USC) group has demonstrated that 3D video conferencing is viable with powerful hardware system configurations (Jones et al., 2009). With these possibilities come the caveat of increased data size through the addition of the third dimension. Previous techniques for 3D video have been proposed (Kauff et al., 2007), but typically involve an intolerance to lossy formats, a lengthy encoding process, or low depth resolution.

Various methods Gumhold et al. (2005); Merry et al. (2006); Schnabel and Klein (2006) have been proposed which allow for amazing compression ratios with lossless quality on 3D point clouds, but these involve lengthy encoding processes. When dealing with real-time 3D video these techniques are undesirable as they do not allow for online encoding of the data, limiting its end-user applications.

The method of compressing a phase map, as indicated in (Jones et al., 2009), at least in structured light scanner cases, comes to mind but has a problem. The depth map is always a floating-point grayscale image, and thus must be packed into a 24-bit image for most image and video compression. This can easily be done by discarding some of the resolution of the floating-point mantissa bits, dropping the 8 least-significant bits, and saving to a lossless format. In this packed format, the phase map can be unpacked and the 3D geometry can be reconstructed with little quality loss. The problem occurs when using lossy formats, such as most video codecs. Since the most significant bits contain the power bits, any change in them changes the unpacked floating-point number drastically.

One solution to this problem is to use depth mapping to encode 3D scenes consisting of unstructured scanned data and structured models into 2D images, and then to rely on 2D image/video compression and transmission techniques. The benefit of doing this is that decades of research and development in 2D image processing can be leveraged utilizing existing compression and transmission techniques along with existing infrastructure. Existing video services such as Youtube and Vimeo can be used with slight modifications; only the video renderer needs to be modified to decode and display 3D scenes rather than 2D images. The Microsoft Kinect is a great example of a real-time 3D scanner that makes use of a depth map. The simple grayscale depth map allows it to reach real-time speeds and leverage 2D image processing. The caveat of their approach is that the depth is limited by the grayscale image color resolution of 256 different values. Also, lossy encoding typically blurs sharp edges in the geometry, and distorts the final rendered scene. These techniques do have the benefit of being extremely fast, through parallelization and simple storage, along with the ability to use existing 2D compression techniques to compress 3D data. Therefore, when we developed the Holovideo compression we designed a technique which uses these benefits.

Holovideo utilizes the basis of the Holoimage technique (Gu et al., 2006) to accomplish the task of depth-mapping an entire 3D scene. Utilizing techniques developed in optical metrology, Holoimage creates a virtual fringe projection system which can depth-map an entire 3D scene into a 2D image. The benefits of such a technique include: (1) using existing research in the field of optical metrology; and (2) leveraging existing research in the field of image processing. However, the original implementation has the limitation of encoding *smooth* geometry because of the nature of the encoding, i.e., continuous 3D surfaces without large step changes or discontinuities. The complexity of the decoding process was also difficult and required a powerful computer and graphics card to rapidly recover 3D geometry for real-time applications.

Recently, we proposed a technique to compress arbitrary 3D shapes using Holoimage Karpinsky and Zhang (2010a). For this technique, an arbitrary 3D shape can be encoded as a 24-bit color image with red and green channel as sine and cosine fringe images, while the blue channel as the stair image for phase unwrapping. Because the third channel is used to unwrap the phase map obtained from red and green channels of fringe images point by point, no spatial phase unwrapping is necessary, thus it can be used to recover arbitrary 3D shapes. However, because a stair image is used for blue channel, any information loss will induce problems to correctly recover 3D geometry, thus the whole color image cannot be stored in any lossy format. This problem becomes more significant for videos because most 2D video formats are inherently lossy.

To circumvent this problems, this paper presents a new coding that encode the blue channel with smoothed cosine functions. Because all three channels use smooth cosine functions, lossy image format can be used to restore the original geometry properly. Because a lossy image format can be used, it enables 3D video encoding with standard 2D video formats. This technique is called Holovideo. The Holovideo technique allows existing 2D video codecs such as QuickTime Run Length Encoding (QTRLE) to be used on 3D videos, resulting in compression ratios of over 134:1 Holovideo to OBJ format. We found that under a compression ratio of 134:1, Holovideo to OBJ file format, the 3D geometry quality drops at a negligible level. Several sets of 3D videos were captured using a structured light scanner, compressed using the Holovideo codec, and then uncompressed and displayed to demonstrate the effectiveness of the codec. With the use of OpenGL Shaders (GLSL), the 3D video codec can encode and decode in realtime. We demonstrated that for a video size of 512×512, the decoding speed is 28 frames per second (FPS) with a laptop computer using an embedded NVIDIA GeForce 9400m graphics processing unit (GPU). Encoding can be done with this same setup at 18 FPS, making this technology suitable for applications such as interactive 3D video games and 3D video conferencing.

Section 6.3 explains the principle of encoding and decoding using OpenGL Shaders (GLSL). Section 6.4 shows experimental results. Section 6.5 discusses the caveats of the proposed technique, and Section 6.6 summarizes this paper.

6.3 Principle

6.3.1 Fringe projection technique

Fringe projection technique is a special structured light method in that it uses sinusoidally varying structured patterns. In a fringe projection system, the 3D information is recovered from *phase* which is encoded naturally into the sinusoidal pattern. To obtain the phase, a phase-shifting algorithm is typically used. Phase shifting is extensively used in optical metrology because of its numerous merits which include the capability to achieve pixel-by-pixel spatial resolution during 3D shape recovery. Over the years, a number of phase-shifting algorithms have been developed including three-step, four-step, least-square algorithms, etc. Malacara (2007). In a real-world 3D imaging system using a fringe projection technique, a three-step phase-shifting algorithm is typically used because of the existence of background lighting and noise. Three fringe images with equal phase shift can be described as

$$I_1(x,y) = I'(x,y) + I''(x,y)\cos(\phi - 2\pi/3), \tag{6.1}$$

$$I_2(x,y) = I'(x,y) + I''(x,y)\cos(\phi), \tag{6.2}$$

$$I_3(x,y) = I'(x,y) + I''(x,y)\cos(\phi + 2\pi/3). \tag{6.3}$$

Where $I'(x,y)$ is the average intensity, $I''(x,y)$ the intensity modulation, and $\phi(x,y)$ the phase to be found. Simultaneously solving Eqs. (6.1)–(6.3) leads to

$$\phi(x,y) = \tan^{-1}\left[\sqrt{3}(I_1 - I_3)/(2I_2 - I_1 - I_3)\right]. \tag{6.4}$$

This equation provides the wrapped phase ranging from 0 to 2π with 2π discontinuities. These 2π phase jumps can be removed to obtain the continuous phase map by adopting a phase-unwrapping algorithm Ghiglia and Pritt (1998). However, all phase-unwrapping

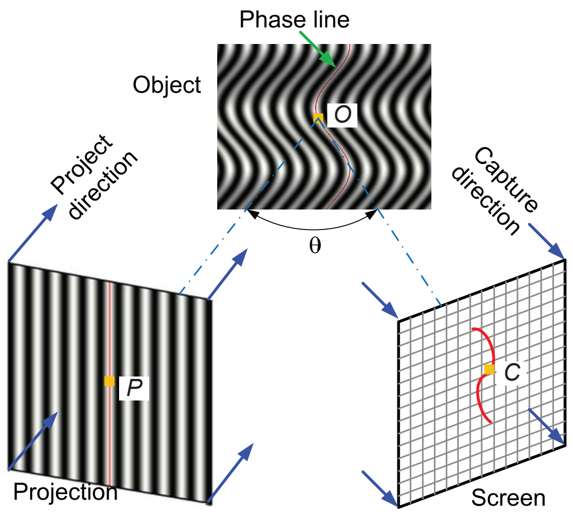


Figure 6.1 Holovideo system setup. The virtual projection system projects sinusoidal fringe patterns onto the object and rendered by the graphics pipeline, and then displayed on the screen. The screen view acts as a virtual camera imaging system. Because both the projector and the camera are virtually constructed, they could be both orthogonal devices. The angle between the projection system and the camera imaging system is θ.

algorithms have the common limitations that they can resolve neither large step height changes that cause phase changes larger than π nor discontinuous surfaces.

6.3.2 Holovideo system setup

The Holovideo technique is devised from the digital fringe projection technique. The idea is to create a virtual fringe projection system, scan scenes into 2D images, compress and store them, and then decompress and recover the original 3D scenes. Holovideo utilizes the basis of the Holoimage technique (Gu et al., 2006) to accomplish the task of depth-mapping an entire 3D scene. Figure 6.1 shows the typical setup of the Holovideo system. The projector projects fringe images onto the object and the camera captures reflected fringe images from another viewing angle. From the camera image, 3D information can be recovered pixel by pixel if the geometric relationship between the projector pixel (P) and the camera pixel (C) is known. Because Holoimage system is precisely defined by the user, both the camera and the projector can be orthogonal devices, their geometric relationship is easy to obtain. Thus, the phase to coordinate conversion is very simple (Karpinsky and Zhang, 2010a). In the virtual fringe projection system, the projector is configured as the projective texture image to project the texture onto the object, the computer screen acts as the camera. The projection angle (θ) is realized by setting the model view matrix of the OpenGL pipeline.

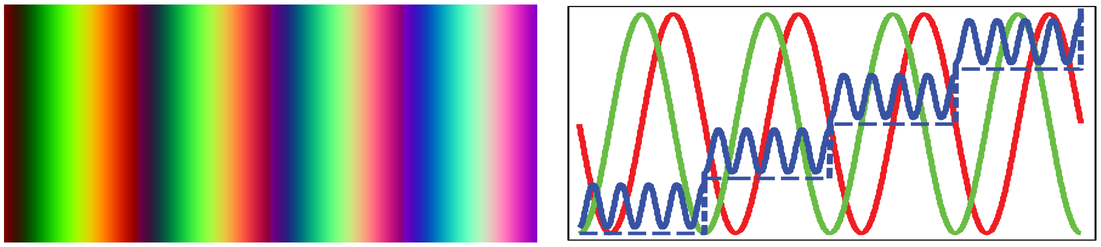
6.3.3 Encoding on GPU

To speed up the encoding process, the Holovideo system was constructed on GPU. The virtual fringe projection system is created through the use of GLSL Shaders which color the 3D scene with the structured light pattern. The result is rendered to a texture, saved to a video file, and uncompressed later when needed. By using a sinusoidal pattern for the structured light system, lossy compression can be achieved without major loss of quality.

As stated before, the Holovideo encoding shader colors the scene with the structured light pattern. To accomplish this, a model view matrix for the projector in the virtual structured light scanner is needed. This model view matrix is rotated around the z axis by some angle ($\theta = 18°$ in our case) from the camera matrix. From here the Vertex Shader can pass the x, y values to the Fragment Shader as a varying variable along with the projector model view, which can then be used to find the x, y values for each pixel from the projectors perspective. At this point, each fragment is colored with the Eqs. (6.5)–(6.7), and the resulting scene is rendered to a texture giving a Holo-encoded scene.

$$\begin{aligned}
I_r(x,y) &= 0.5 + 0.5\sin(2\pi x/P), && (6.5)\\
I_g(x,y) &= 0.5 + 0.5\cos(2\pi x/P), && (6.6)\\
I_b(x,y) &= S \cdot Fl(x/P) + S/2 + (S-2)/2 \cdot \cos[2\pi \cdot Mod(x,P)/P_1], && (6.7)
\end{aligned}$$

Here P is the fringe pitch, the number of pixels per fringe stripe, $P_1 = P/(K+0.5)$ is the



(a) (b)

Figure 6.2 The encoded structured pattern. (a) The structured pattern, whose three channels are all encoded with cosine functions. (b) One cross section of the structured pattern. Note that all channels use cosine waves to reduce problems associated with lossy encoding.

local fringe pitch and K is an integer number, S is the stair height in grayscale intensity value, $Mod(a, b)$ is the modulus operator to get a over b, and $Fl(x)$ is to get the integer number of x by removing the decimals. Figure 6.2 illustrates a typical structure pattern for Holovideo. Compared to previous work Karpinsky and Zhang (2010a) Equation 6.7 has been modified to vary sinusoidally. This is done so that all three color channels can be stored in a lossy format, which is addressed later during the experimental results discussion.

After each render, which renders to a texture, we pull the texture from the GPU and save it as a frame in the current movie file. The two main bottlenecks are transferring all of the geometry to the graphics card to be encoded, and copying the resulting texture from the graphics card to the movie file in the computer memory. Since we already have to transfer the geometry to the GPU there is nothing we can do about the former bottleneck. The latter bottleneck, however, can be mitigated by accessing textures from the GPU through DMA using pixel buffer objects, resulting in asynchronous transfers.

6.3.4 Decoding on GPU

Decoding the resulting Holovideo is more involved than encoding, as there are more steps, but it can be scaled to the hardware by simply subsampling. In decoding, four major steps need to be accomplished: (1) calculating the phase map from the Holovideo frame, (2) filtering the phase map, (3) calculating normals from the phase map, and (4) performing the final render. To accomplish these four steps, we utilized multipass rendering, saving results from the intermediate steps to a texture, which allowed us to access neighboring pixel values in proceeding steps.

To calculate the phase map, we set up the rendering with an orthographic projection and a render texture and then rendered a screen-aligned quad. With this setup, we can perform image processing using GLSL. From here, the phase-calculating shader took each pixel value and applied Eq. (6.8) below, saving the result to a floating-point texture for the next step in the pipeline. Equations (6.5)–(6.7) provide the phase uniquely for each point.

$$\Phi(x, y) = 2\pi \times Fl[(I_b - S/2)/S] + \tan^{-1}[(I_r - 0.5)/(I_g - 0.5)]. \tag{6.8}$$

Unlike the phase obtained in Eq. (6.4) with 2π discontinuities, the phase obtained here is already unwrapped naturally without the common limitations of conventional phase unwrapping algorithms. Therefore, it can be used to encode an arbitrary 3D scene scanned by a 3D scanner even with step height variations. It is important to notice that under the virtual fringe projection system all lighting can be controlled or eliminated, thus the phase can be obtained by two-channel fringe patterns with $\pi/2$ phase shift. This allows for the third channel to be used for phase unwrapping.

Since the phase is calculated point by point, it allows for leveraging the parallelism of the GPU for the decoding process. It is also important to notice that instead of directly using the stair image as proposed in Reference Karpinsky and Zhang (2010a), we use a cosine function to represent this stair image as described by Eq. (6.7). If the image is stored in a lossy format, the smooth cosine function causes less problems than the straight stair function with sharp edges.

From the unwrapped phase $\Phi(x, y)$ obtained in Eq. (6.8), the normalized coordinates

(x^n, y^n, z^n) can be decoded as Karpinsky and Zhang (2010a)

$$x^n = j/W, \tag{6.9}$$

$$y^n = i/W, \tag{6.10}$$

$$z^n = \frac{P\Phi(x,y) - 2\pi i \cos(\theta)}{2\pi W \sin\theta}. \tag{6.11}$$

This yields a value z^n in terms of P which is the fringe pitch, i, the index of the pixel being decoded in the Holovideo frame, θ, the angle between the capture plane and the projection plane ($\theta = 18°$ for our case), and W, the number of pixels horizontally.

From the normalized coordinates (x^n, y^n, z^n), the original 3D coordinates can recovered point by point

$$x = x^n \times S_c + C_x, \tag{6.12}$$

$$y = y^n \times S_c + C_y, \tag{6.13}$$

$$z = z^n \times S_c + C_z. \tag{6.14}$$

Here S_c is the scaling factor to normalize the 3D geometry, (C_x, C_y, C_z) are the center coordinates of the original 3D geometry.

Because of the subpixel sampling error, we found that some areas of the phase $\Phi(x, y)$ have one-pixel jumps along the edge of the stair image on I_b. This problem can be easily filtered out since it is only one pixel wide. The filter that we perform on the phase map is a median filter which removes spiking noise in the phase map. We used McGuire's method, allowing for a fast and efficient median filter in a GLSL Shader McGuire (2008).

Normal calculation is done by calculating surface normals with adjacent polygons, and then averaging them together to form a normal map. Again, this uses the same setup as above with the orthogonal projection, render texture, and screen-aligned quad.

At last we have the final render step. Before we perform this step, we switch to a perspective projection, although an orthogonal projection could be used. We also bind the back screen buffer as the main render buffer, bind the final render shader, and then render a plane of pixels. With the plane of pixels, we can reduce the number of vertices by some divisor of the width and height of the Holovideo. This allows us to easily subsample the Holovideo, reducing the detail of the final rendering but also reducing the computational load. This is what allows the Holovideo to scale from devices with small graphics cards to those with large workstation cards.

6.3.5 3D video compression

Because each frame is encoded with cosine functions, lossy image formats can be used. Therefore, lossy compression results in little loss of quality if the codec is properly selected. Most codecs use some transform that approximates the Karhunen-Loève Transform (KLT) such as the cosine or integer transform. These transforms work the best on so-called natural images where there are no sharp discontinuities in the color space of the local block that the transform is applied to. Since the Holovideo uses cosine waves, the discontinuities are minimized and the transform yields highly compressed blocks which can then be quantized and encoded.

6.4 Experimental results

To verify the performance of the proposed Holovideo encoding system, we first encode one single 3D frame with rich features. Figure 6.3 shows the results. For this example, the Holovideo system is configured as follows: The image resolution is $512(W) \times 512(H)$; The angle between the projection and the camera $\theta = 18°$; The fringe pitch $P = 32$ pixels; The high-frequency modulation pitch $P = 6$ pixels; And the stair height is $S = 16$. Figure 6.3(a) shows the original 3D geometry that is compressed into a single color Holoimage as shown in Fig. 6.3(b). The red and green channels are encoded as sine and cosine fringe patterns as shown in Fig. 6.3(c) and Fig. 6.3(d), respectively. Figure 6.3(e) shows the blue channel image that is compose of stair with high-frequency cosine func-

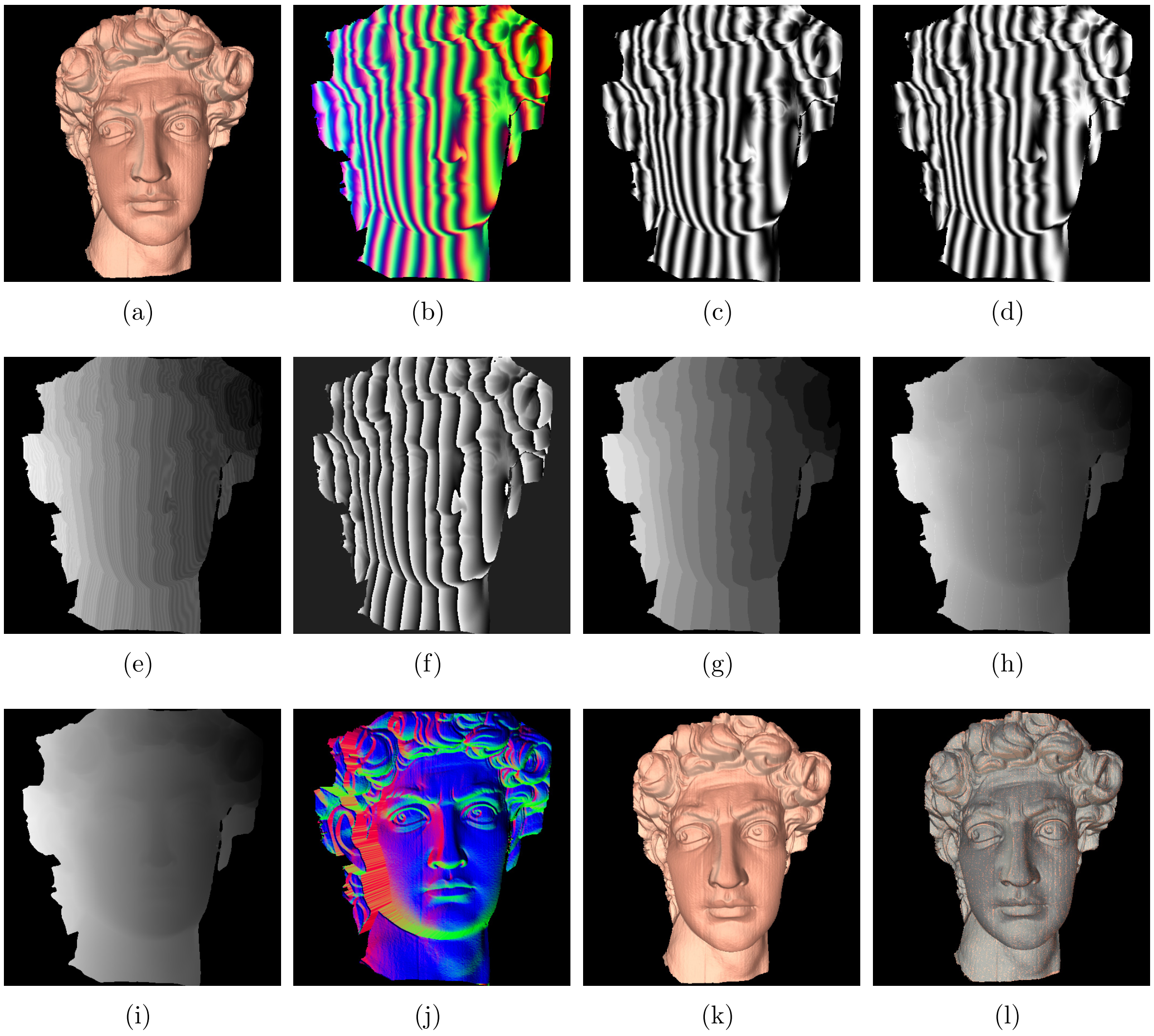


Figure 6.3 Example of the Holovideo codec encoding a single frame from 3D to 2D and then decoding back to 3D. (a) The original scanned 3D geometry by a structured light scanner; (b) The encoded 3D frame into 2D image; (c)-(e) Three color channels of the encoded 2D image frame; (f) Wrapped phase from red and green channel fringe patterns; (g) Image codec used for unwrapping the wrapped phase point by point; (h) The unwrapped phase map using Eq. (6.8); (i) The unwrapped phase after filtering; (j) The normal map; (k) The final 3D recovered geometry; (l) Overlapping the original 3D geometry with the recovered one.

tions following Eq. (6.7). From the red and green channels, the phase can be wrapped with a value ranging from $-\pi$ to $+\pi$ as shown in Fig. 6.3(f). From the third channel, a stair image shown in Fig. 6.3(g) can be obtained, which can be used to unwrap the phase. Figure 6.3(h) shows the unwrapped raw phase map $\Phi(x, y)^r$. Because of the sub-pixel sampling issue, the perfectly designed stair image and the wrapped phase may have misalignment on edges. This figure shows some artifacts (white does) on the raw unwrapped phase map.

Because the artifacts are single pixel in width, they could be removed by applying a median filter to obtain smoothed unwrapped phase $\Phi^s(x, y)$. However, applying a single median filter will make the phase on those artifact point incorrect. Fortunately, because the phase changes must be multiples $(n(x, y)$ of 2π for those artifact points, we only need to determine the integer number $n(x, y)$ to correct those points. In this research, $n(x, y)$ was determined as

$$n(x, y) = \text{Round}\left[\frac{\Phi^s(x, y) - \Phi^r(x, y)}{2\pi}\right], \tag{6.15}$$

and the correctly unwrapped phase map can be obtained by

$$\Phi(x, y) = \Phi(x, y)^r + n(x, y) \times 2\pi. \tag{6.16}$$

Figure 6.3(i) shows the unwrapped phase map after properly removing those artifacts using the aforementioned procedures. The normal map can be calculated once the (x, y, z) coordinates could be calculated using Eqs. (6.12)-(6.14). Figure 6.3(j) shows the computed normal map. Finally, 3D geometry could be rendered on GPU as shown in Fig. 6.3(k). To compare the precision of the reconstructed 3D geometry and the original 3D geometry, they are rendered in the same scene as shown in Fig. 6.3(l), here gold color geometry represents the original 3D geometry and the gray color represents the recovered 3D geometry. It clearly shows that they are well aligned together, which conforms to our previous finding: the error is negligible for the Holoimage to represent a 3D geometry Zhang and Yau (2008).

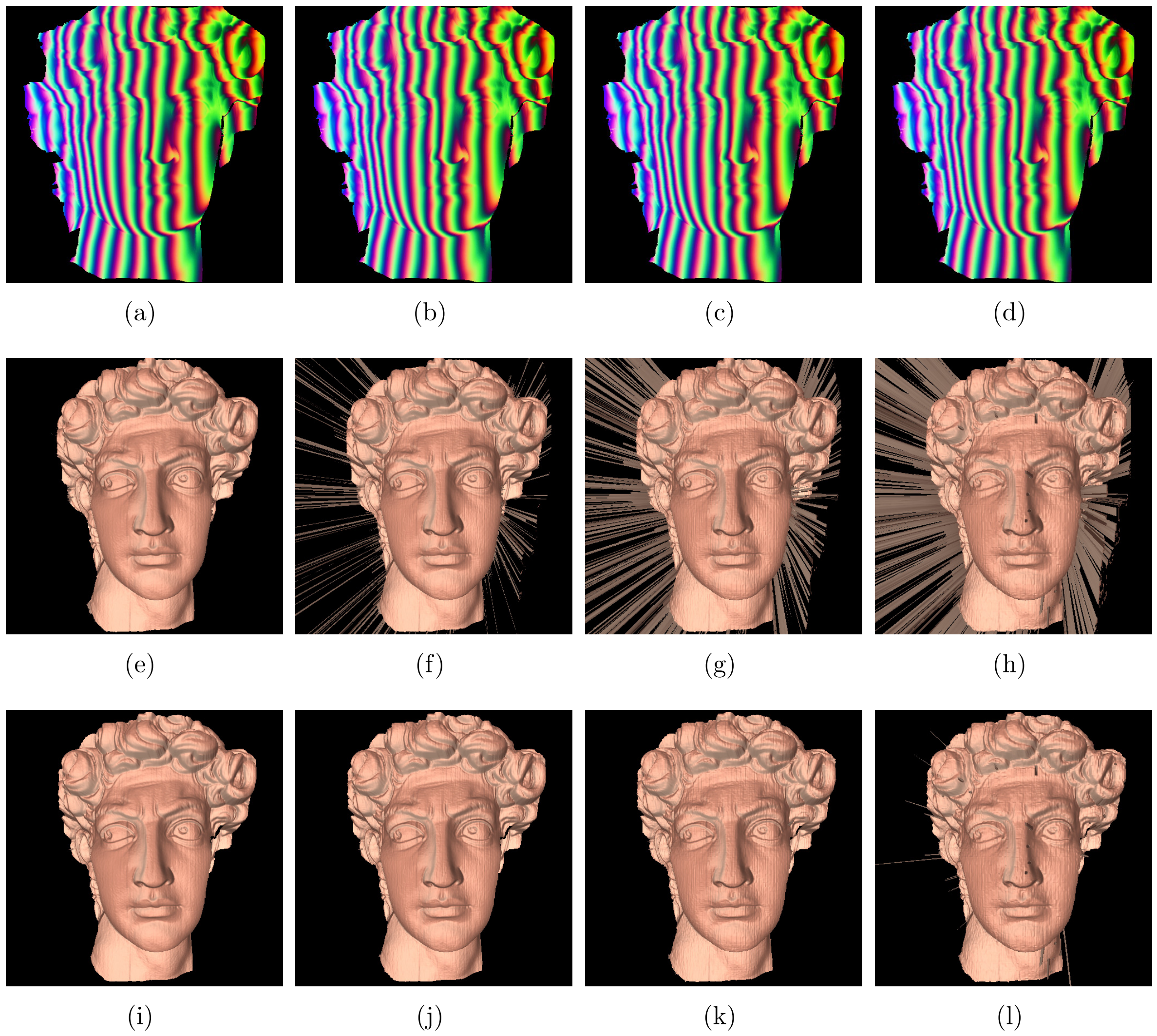


Figure 6.4 The effect of compressing an individual frame with a lossy JPEG file format. (a)-(d) The Holovideo encoded compressed frame with different compression ratios; (e)-(h) The corresponding decoded 3D geometry from the above images. (i)-(l) The 3D geometry of above row after boundary cleaning.The images in (a)-(d) respectively show the compression ratios of 104:1, 174:1, 237:1, and 310:1 when compared against the OBJ file format.

To demonstrate the potential of compressing Holovideo with lossy formats, we compressed a single frame with varying levels of JPEG compression under Photoshop 10.0. Figure 6.4 shows the results of compressing the 3D frame shown Figure 6.3. Figures 6.4(a)-6.4(d) shows compressed JPEG images with the quality levels of 12, 10, 8, 6, respectively. From these encoded images, the 3D shape can be recovered, as shown in Figs. 6.4(e)-6.4(h). It can been that the image can be stored as high quality lossy JPEG files without bringing obvious problems to recovered 3D geometry. With more compressed images being used, the recovered 3D geometry quality reduces (i.e., losing details), and some artifacts (spikes) start appearing. However, most of problematic points occur around boundary regions which are caused by sharp intensity changes of the image. The boundary problems can be significantly reduced if a few pixels are dropped out. Figures 6.4(i)-6.4(l) show the corresponding results after removing the boundary. This experiment clearly indicates that the proposed encoding method allows the use of the lossy imaging format.

OBJ file format is widely used to store 3D mesh data. If the original 3D data is stored in OBJ format without normal information, the file size is 20,935 KB. In comparison with the OBJ file format, we could reach 174:1 with slight quality dropping. When the compression ratio reaches 310:1, the quality of 3D geometry is noticeability reduced, but the overall 3D geometry is still well preserved.

As a comparison, if we use the encoding method discussed in Reference Karpinsky and Zhang (2010a) with blue channels as a straight stair function. The encoded image is shown in Fig. 6.5(a). If the image is stored as lossless format (e.g., bitmap), the 3D geometry can be accurately recovered as illustrated in Fig.6.5(b). Figure 6.5(c) shows that even the image is stored as the highest quality level (12) JPEG format, the 3D recovered result appears to be problematic. If the image quality is reduced to quality level 10, the 3D shape cannot be recovered, as show in Fig. 6.5(d). It is important to note that the boundaries of 3D recovered results were cleaned using the same aforementioned approach. This clearly demonstrate that the previously proposed encoding method cannot be used if a lossy image format is needed.

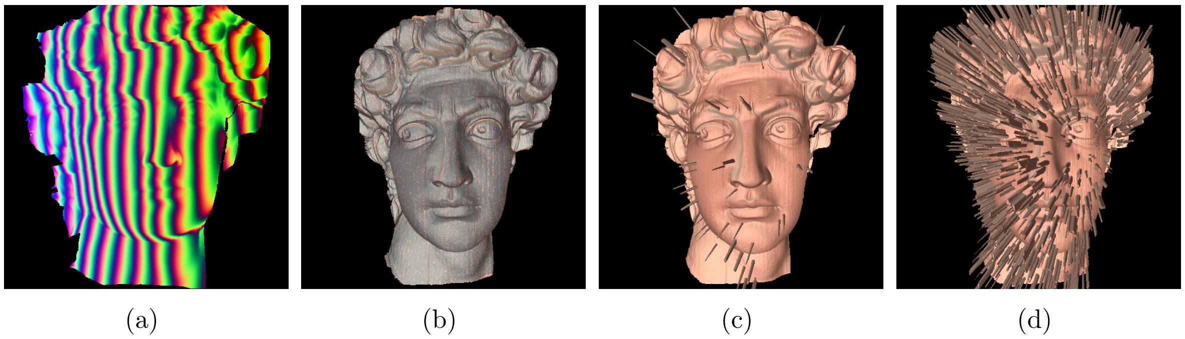


Figure 6.5 Comparing results of storing prior encoded image with a lossy JPEG file format. (a) The encoded Holovideo frame using the method introduced in Reference (Karpinsky and Zhang, 2010a); (b) Overlap the original 3D scanned data with the recovered 3D geometry from the lossless bitmap file; (c) 3D recovered shape when when the image was stored as lossy JPEG format with quality level 12; (d) 3D recovered shape when when the image was stored as lossy JPEG format with quality level 10.

To show that the proposed method can be used to encode and decode 3D videos, we captured a short 45-second clip of an actress using a structured light scanner (Zhang and Yau, 2006) running at 30 FPS with an image resolution of 640×480 per frame. Saving each frame out in the OBJ format, we end up with over 42GB worth of data. Then we took the data, ran it through the Holovideo encoder, and saved the raw lossless 24-bit Bitmap data to an AVI file with a resolution of 512×512, which resulted in a file that was 1 GB. This is already a compression of over 42:1 Holovideo to OBJ. Next we JPEG-encoded each frame and ran it through the QTRLE codec, which resulted in a file that was 314.3 MB, achieving a ratio of over 134:1 Holovideo to OBJ. Media 1 associated with Fig. 6.6 shows the original scanned 3D video (the video on the left), the encoded Holovideo (the video in the middle), and the decoded 3D video (the video on the right). The resulting video had no noticeable artifacts from the compression, and it could be further compressed with some loss of quality. All of the encoding and decoding processes are performed on the GPU in real-time (28 FPS decoding and 18 FPS encoding) with

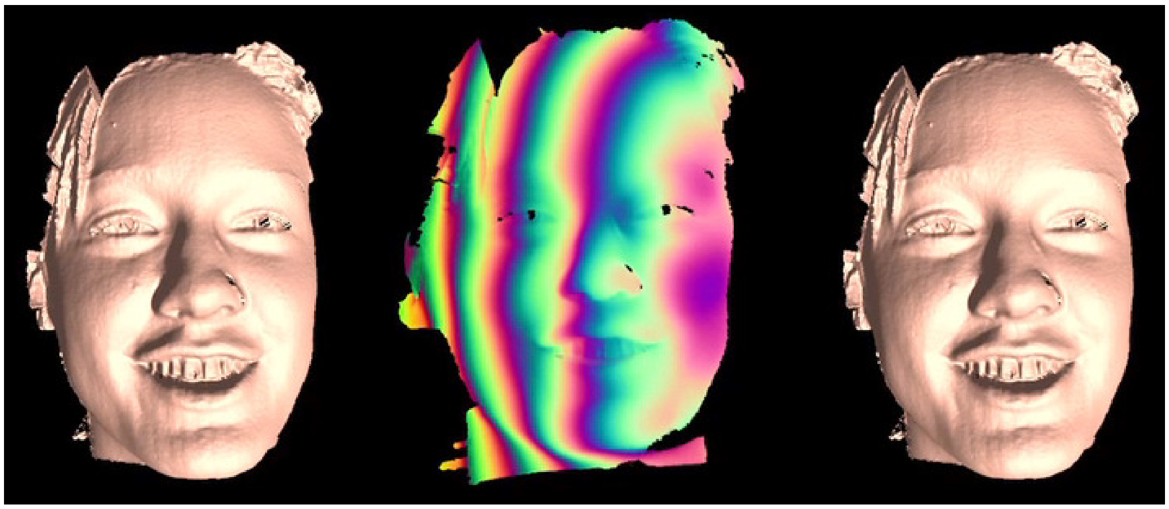


Figure 6.6 3D video compression result using the proposed Holovideo technique (Media 1). The left video shows the original scanned 3D video, the middle video shows the encoded Holovideo, and the right video shows the decoded 3D video.

a simple graphics card (NVIDIA GeForce 9400m) and the resulting compression ratio is over 134:1 in comparison with the 3D data stored in OBJ file format.

6.5 Discussion

One caveat to note is that a lot of video codecs are tailored to the human eye and reduce information in color spaces that humans typically do not notice. An example of this is the H.264 codec which converts source input into the YUV color space. The human eye has a higher spatial sensitivity to luma (brightness) than chrominance (color). Knowing this, bandwidth can be saved by reducing the sampling accuracy of the chrominance channels with little impact on human perception of the resulting video.

Compression codecs that use the YUV color space currently do not work with the Holovideo compression technique as they result in large blocking artifacts. Thus we used the QTRLE codec which is a lossless run length encoding video codec. To achieve lossy video compression, we JPEG-encoded the source images in the RGB color space and then passed them to the video encoder. This allows us to achieve a high compression ratio at a controllable quality level. Future work will entail researching different fringe patterns which fit into the YUV color space.

6.6 Conclusion

We have presented a technique which can encode and decode high-resolution 3D data in real-time, thus achieving 3D video. Decoding was performed at 28 FPS and encoding was performed at 17 FPS on an NVIDIA GeForce 9400m GPU. Due to the design of the algorithm, standard 2D video codecs can be applied so long as they can encode in the RGB color space. Our results showed that a compression ratio of over 134:1 can be achieved in comparison with the OBJ file format. By using 2D video codecs to compress the geometry, existing research and infrastructure in 2D video can be leveraged in 3D.

Acknowledgements

Thank you to Amy Dixon for staring in the video, Ying Xu for performing scans, Laura Ekstrand for critiquing the writing, and the other members of the 3D Machine Vision Lab at Iowa State university for their help and suggestions.

CHAPTER 7. 3D range geometry video compression with the H.264 codec

A paper published in the Journal of Optics and Lasers in Engineering in May 2013.[1]

Nikolaus Karpinsky[2,4] and Song Zhang[3,4]

7.1 Abstract

Advances in three-dimensional (3D) scanning have enabled the real-time capture of high-resolution 3D videos. With these advances brings the challenge of streaming and storing 3D videos in a manner that can be quickly and effectively used. This research addresses this challenge by generalizing the Holovideo technique to video codecs that use the YUV colorspace such as the H.264 codec. With the H.264 codec, we have achieved a compression ratio of over 6086 : 1 (Holovideo to OBJ) with a reasonably high quality; utilizing an NVIDIA GeForce 9400m GPU, we have realized 17 frames per second encoding, and 28 frames per second decoding speed, making it a viable solution for real-time 3D video compression.

[1]This paper was published in the Journal of Optics and Lasers in Engineering and is made available as an electronic reprint with the permission of Elsevier. The paper can be found at the following URL on the Elsevier website: http://www.sciencedirect.com/science/article/pii/S0143816613000067. Systematic or multiple reproduction of distribution to multiple locations via electronic or other means is prohibited and is subject to penalties under law.

[2]Primary researcher and author

[3]Assistant professor and advisor.

[4]Department of Mechanical Engineering, Iowa State University

7.2 Introduction

Advances in 3D scanning have enabled the real-time capture of high-resolution 3D video. These advances have brought forth the challenge of streaming and storing these high-resolution 3D video frames in a format that can be quickly and efficiently used. Classical approaches in 3D geometry compression compress the 3D coordinates and their attributes such as normals, UV coordinates, etc. in a model format such as OBJ, PLY, STL. Although these formats work well for static scans or structured meshes with pre-defined animation, the same does not hold true for high-resolution 3D video frames due to their unstructured animation nature.

To deal with this challenge, different approaches have been taken such as heuristic based encoding of 3D point clouds (Merry et al., 2006; Gumhold et al., 2005), and image based encoding approaches (Gu et al., 2002; Krishnamurthy et al., 2001; Gu et al., 2006). Image based encoding approaches work well, as the 3D geometry can be projected into 2D images and then compressed using 2D image compression techniques. Since 2D image compression is a long studied field, high compression ratios can be achieved with little loss of quality. Later when the 3D geometry is needed, it can be recovered from the 2D images using image based rendering. There are three key steps to effectively compressing the geometry with these techniques; (1) projecting the 3D geometry into 2D images, (2) correctly encoding and decoding the projected images with a 2D codec, (3) recovering the 3D geometry from the 2D images.

This research addresses the second key step, correctly encoding and decoding the projected images with a 2D codec. Typically, 2D video codecs are tailored to natural sinusoidally varying images with color redundancies between frames. The codecs are tailored to natural sinusoidally varying images with the transform that they use, such as the discrete cosine transform or the integer transform. These transforms are applied to image blocks and then small variations are quantized off resulting in slightly lossy encoding at high compression levels. Detecting and encoding changes between frames instead of repeating nearly redundant information leverages color redundancies between frames. With this encoding certain frames are stored (keyframes) with changes being applied to recover frames between the stored frames (interframes) (Richardson, 2010).

Previous research has shown that the Holoimage technique (Gu et al., 2006; Karpinsky and Zhang, 2010a) can be extended to 3D video by modifying the fringe equations and then using OpenGL Shaders and asynchronous direct memory access (DMA) transfers to the graphics processing unit (GPU) (Karpinsky and Zhang, 2012b). This research used JPEG encoding on the frames and then used Quicktime Run Length Encoding on each of the frames to achieve a compressed 2D representation of the 3D geometry. With this encoding, compression ratios of over 134:1 Holovideo frame to OBJ can be achieved, at 17 frames per second encoding with an NVIDIA GeForce 9400m GPU. Although good compression is achieved with no noticeable artifacts, it is not optimal as JPEG encoding in the RGB color space with Quicktime Run Length Encoding is not a standard 2D video encoding technique.

This research addresses this by extending the Holovideo technique to the H.264 codec, which is a standard 2D video codec. By applying a conversion to the planar YUV444 and YUV422 color formats, the Holovideo frames can be encoded with the H.264 encoder. With this encoding, compression ratios of over $352:1$ when compared to the OBJ file format have been achieved with a mean squared error as low as .204%.

Section 7.3 explains the principle behind the technique, addressing encoding, compressing, and decoding using OpenGL Shaders (GLSL) and the H.264 codec. Section 7.4 shows experimental results with a unit sphere and short 45 second 3D video, and Section 7.5 summarizes the paper.

7.3 Principle

7.3.1 Fringe projection technique

The fringe projection technique is a structure light method from optical metrology that uses sinusoidally varying structured light patterns. 3D information is recovered from *phase* which is encoded in the sinusoidal pattern. To obtain the phase from the recovered images, a phase-shifting algorithm is typically employed. Phase shifting is used because of its numerous merits, including the capability to achieve pixel-by-pixel spatial resolution during 3D shape recovery. Over the years, a number of phase-shifting algorithms have been developed including three-step, four-step, least-square algorithms, etc. (Malacara, 2007).

In a real-world 3D imaging system making use of a fringe projection technique, a three-step phase-shifting algorithm is typically employed due to its ability to help reduce background lighting and noise while using a small number of fringe patterns. Three fringe images with equal phase shift can be described with the following equations

$$I_1(x,y) = I'(x,y) + I''(x,y)\cos(\phi - 2\pi/3) \tag{7.1}$$

$$I_2(x,y) = I'(x,y) + I''(x,y)\cos(\phi) \tag{7.2}$$

$$I_3(x,y) = I'(x,y) + I''(x,y)\cos(\phi + 2\pi/3) \tag{7.3}$$

where $I'(x,y)$ is the average intensity, $I''(x,y)$ the intensity modulation, and $\phi(x,y)$ the phase to be found. Simultaneously solving Eqs. (7.1)–(7.3) leads to

$$\phi(x,y) = \tan^{-1}\left[\sqrt{3}(I_1 - I_3)/(2I_2 - I_1 - I_3)\right] \tag{7.4}$$

This equation yields the wrapped phase $\phi(x,y)$ ranging from 0 to 2π with 2π discontinuities. A conventional phase-unwrapping algorithm can be adopted to remove these 2π phase jumps and obtain a continuous phase map (Ghiglia and Pritt, 1998). This algorithm simply traverses the wrapped phase map adding integer values of 2π to $\phi(x,y)$,

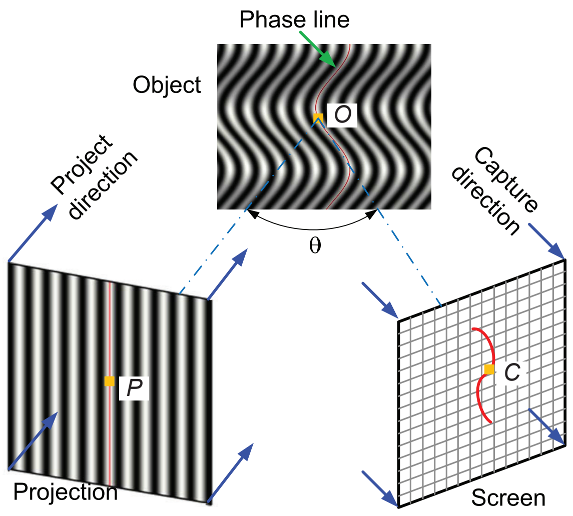


Figure 7.1 Holovideo system conceptual model. The virtual projection system projects sinusoidal fringe patterns onto the object, the result is rendered by the graphics pipeline, and then displayed on the screen. The screen view acts as a virtual camera imaging system. Because both the projector and the camera are virtually constructed, they can both be orthogonal devices. The angle between the projection system and the camera imaging system is θ.

which can be modeled with the following equation

$$\Phi(x, y) = \phi(x, y) + k \times 2\pi \tag{7.5}$$

where $\phi(x, y)$ is the wrapped phase, k is the integer number of phase jumps, and $\Phi(x, y)$ is the unwrapped phase. However, all conventional phase-unwrapping algorithms suffer from the limitations that they can neither resolve large step height changes that cause phase changes larger than π nor discontinuous surfaces.

7.3.2 Holovideo system setup

The Holovideo technique is a specialized fringe projection technique that uses a virtual fringe projection system. This virtual fringe projection system scans 3D scenes into 2D images, compresses and stores them, and then decompresses and recovers the original 3D scenes. Holovideo utilizes the Holoimage technique (Gu et al., 2006) to depth map 3D scenes into 2D images. Figure 7.1 shows a conceptual model of the Holovideo system. In this model, the projector projects fringe images onto the scene and the camera captures the reflected fringe images from another angle. The projector in this conceptual model can be realized as a projective texture implemented through the use of the OpenGL Shading Language (GLSL), and the camera can be realized as the framebuffer. From the camera image, 3D information can be recovered pixel-by-pixel if the geometric relationship between the projector pixel (P) and the camera pixel (C) is known. Since the Holoimage system is mathematically defined using a computer graphics pipeline, both the camera and projector can be orthogonal devices and their geometric relationship can be precisely defined. Thus, converting from phase to 3D coordinates is very simple and can be done in parallel (Karpinsky and Zhang, 2010a).

7.3.3 Encoding

To encode the 3D scene, the Holovideo system uses the virtual fringe projection system, which is created through the use of OpenGL Shaders. These shaders color the 3D scene with a structured light pattern defined by the following equations.

$$I_r(x,y) = 0.5 + 0.5\sin(2\pi x/P), \tag{7.6}$$

$$I_g(x,y) = 0.5 + 0.5\cos(2\pi x/P), \tag{7.7}$$

$$I_b(x,y) = S \cdot Fl(x/P) + S/2 + (S-2)/2 \cdot \cos[2\pi \cdot Mod(x,P)/P_1], \tag{7.8}$$

Here P is the fringe pitch, the number of pixels per fringe stripe, $P_1 = P/(K+0.5)$ is the local fringe pitch, K is an integer number, S is the stair height in grayscale intensity value, $Mod(a,b)$ is the modulus operator to get a over b, and $Fl(x)$ is the floor function to get the integer number of x. The depth resolution is then dependent on the pitch P with smaller pitches reducing quantization noise but increasing spiking noise; a discussion of the effects along with optimal values can be found in previous work (Karpinsky, 2011). Figure 7.2 illustrates a typical structure pattern for Holovideo with (a) showing the resulting pattern rendered as an RGB image, and (b) a cross section of the pattern with

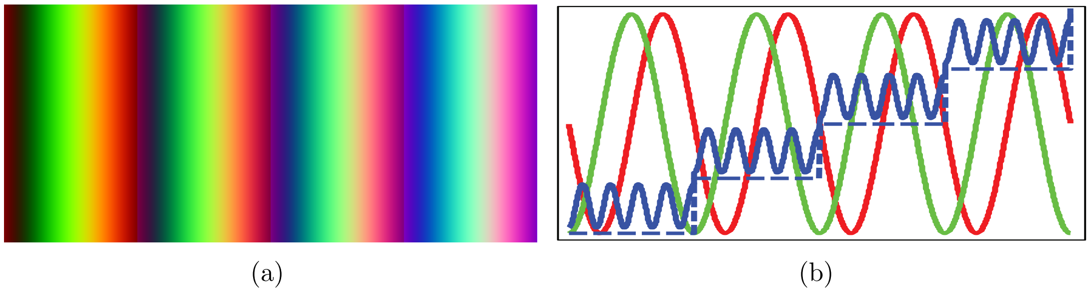


Figure 7.2 The encoded structured pattern used by the Holovideo technique. (a) The structured pattern displayed as an RGB image. (b) A graph of one cross section of the structured pattern. Note that all channels use cosine waves to reduce problems associated with lossy encoding.

the intensity of each color channel graphed.

For the Holovideo encoding vertex shader, a model view matrix for the projector and for the camera in the virtual structure light scanner is needed. The model view matrix for the projector is rotated around the z axis by some angle ($\theta = 30^{\circ}$ in our case) from the camera model view matrix. From here the vertex shader passes the x, y values to the fragment shader as a varying variable along with the projector model view matrix, so that x, y values for each pixel can be determined from the projectors perspective. In the fragment shader, each fragment is colored with the Eqs. (7.6)–(7.8), and the resulting scene is rendered to a texture yielding a Holovideo encoded frame. It is important to notice that instead of directly using the stair image as proposed in Reference (Karpinsky and Zhang, 2010a), a cosine function is used to represent this stair image as described by Eq. (7.8).

Each frame of the 3D video is rendered to a texture in this fashion, and then the resulting texture is pulled from the GPU to the CPU where it can be transformed and then passed to a 2D video codec. To mitigate the bottleneck of transferring the textures from the GPU to the CPU, asynchronous DMA transfers are employed using pixel buffer objects (PBOs).

7.3.4 Video compression

One of the challenges in directly taking Holovideo frames into H.264 is that most H.264 codecs work in the YUV color space. If the frames are directly passed into the codec, it will convert the RGB Holovideo frame to a planar YUV frame and then compress it. Coming back out, the information is decompressed and converted back into RGB. If this process is done to the Holovideo frame Figure 7.3 (a), large errors are introduced, shown in Figure 7.3 (b). One way to address this issue would be to encode three separate movies where in each movie the YUV components would correspond to a specific color channel RGB. This would then allow for the data to be compressed and then later decompressed and reconstructed, but would require three separate synced video streams. Instead, addressing this in a better way, we transform the Holovideo frame directly into the planar YUV color space with the step height channel in the Y component, and the fringe in the U and V, shown by Figure 7.4 (a). Then the H.264 codec can directly compress these frames with little loss of error shown in Figure 7.4 (b).

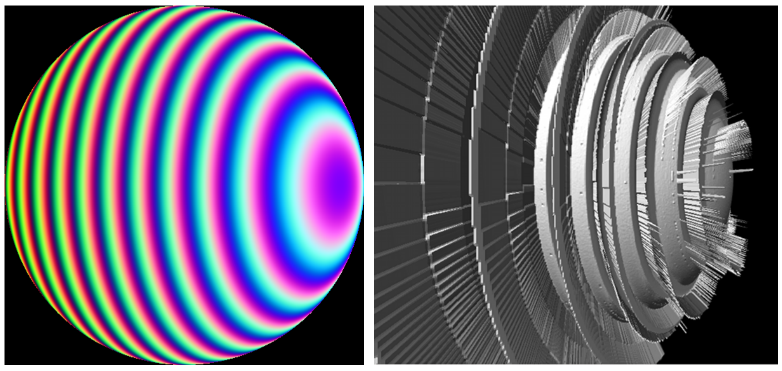

(a) (b)

Figure 7.3 Holovideo encoded unit sphere with H.264 encoding. (a) RGB Holovideo frame of unit sphere directly encoded by H.264. (b) Reconstructed unit sphere with Holovideo frame from (a).

Another challenge associated with H.264 video encoding is downsampling, which occurs with the frames. Since the human eye is less sensitive to color variations (chrominance UV) versus intensity variations (luminance Y), downsampling of the UV components is typically employed. The H.264 codec supports this by downsampling the UV components with YUV422 or YUV420 encoding. In this encoding scheme each pixel has an intensity or Y component, but chrominance UV components are shared between pixels. This reduces the overall bit rate with some lossy error being introduced. Downsampling with YUV422 encoding on a Holovideo frame is shown with Figure 7.4 (c).

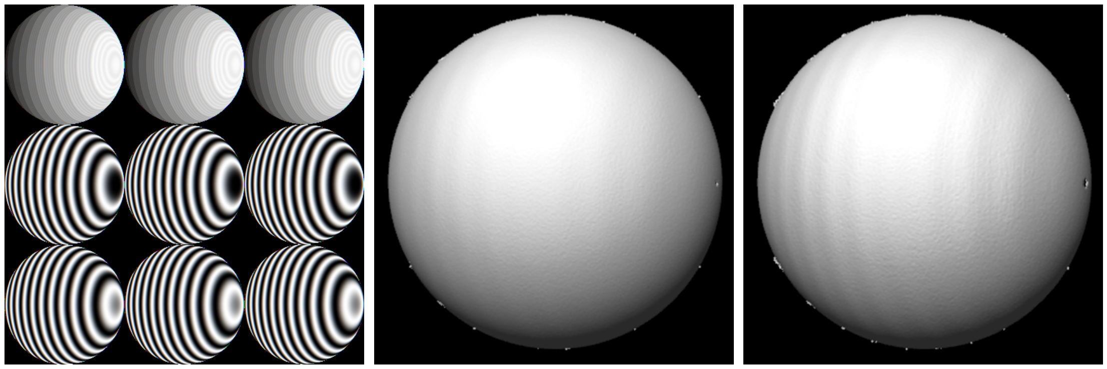


Figure 7.4 Transformed Holovideo encoded unit sphere with H.264 encoding. (a) Planar YUV Holovideo frame of a unit sphere encoded by H.264. (b) Reconstructed unit sphere if Holovideo frame is encoded from YUV444 into H.264. (c) Reconstructed unit sphere if Holovideo frame is encoded from YUV422 into H.264.

7.3.5 Decoding on GPU

Decoding a Holovideo frame is a more involved process than encoding, as there are more steps requiring multi-pass rendering, but the process scales with the hardware through subsampling. In the decoding pipeline, shown in figure 7.5, the five major steps that need to be performed are: (1) calculating an unwrapped phase map from the Holovideo frame, (2) filtering the unwrapped phase map, (3) calculating a floating point depth map from the filtered unwrapped phase map, (4) calculating normals from the floating point depth map, (5) performing the final render. To accomplish these five steps, multi pass rendering can be utilized, saving the results from each pass to a texture which allows neighboring pixel value access during the proceeding steps.

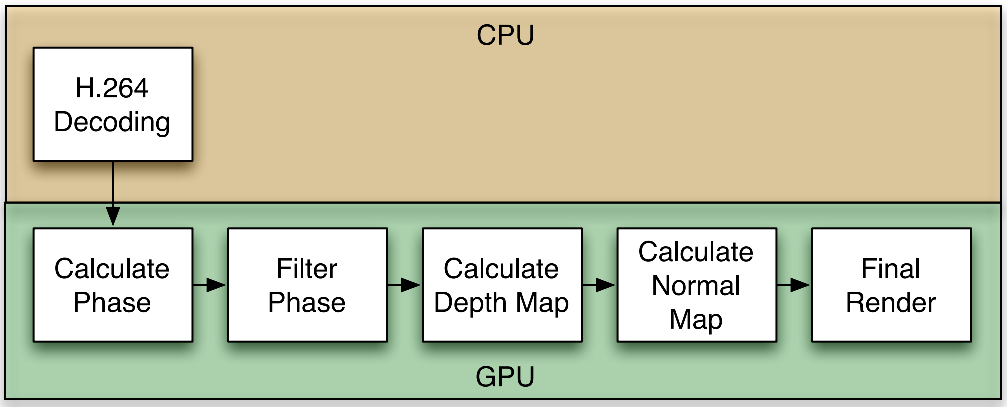


Figure 7.5 Holovideo decoding pipeline. First, frames are decoded using the H.264 decoder on the CPU side, and frames are passed to the GPU as textures. Next from the Holovideo frame, the unwrapped phase map is calculated and filtered, followed by calculating a depth map and normal map. Finally, final rendering is performed with the depth map and normal map, resulting in the reconstructed scene.

During steps (1) - (4) an orthographic projection with a screen aligned quad and render texture the size of the Holovideo frame is used to perform image processing. Each input image is entered into the shaders through a texture, the vertex shader simply passes the four vertices through, and then the fragment shader performs the pixel wise image processing.

To calculate the unwrapped phase map, step (1), we input the Holovideo frame and apply Eq. (7.9) below, saving the resulting unwrapped phase map to a floating point texture for the next step in the pipeline. Equations (7.6)–(7.8) provide the phase uniquely for each point.

$$\Phi(x, y) = 2\pi \times Fl[(I_b - S/2)/S] + \tan^{-1}[(I_r - 0.5)/(I_g - 0.5)]. \tag{7.9}$$

Unlike the phase obtained in Eq. (7.4) with 2π discontinuities, the phase obtained here is already unwrapped without the limitations of conventional phase unwrapping algorithms. Therefore, scenes with large height variations can be encoded which is not true when using conventional phase unwrapping algorithms. It is also important to notice that under the virtual fringe projection system, all lighting is eliminated, thus the phase can be obtained by using only two fringe patterns with $\pi/2$ phase shift. This allows for the third channel to be used for phase unwrapping.

For step (2) we used a modified median filter similar to the one proposed by McGuire (McGuire, 2008). The reason that median filtering needs to be applied is due to sub-pixel sampling and quantization errors during 2D image compression. Some areas of the phase $\Phi(x, y)$ have one-pixel jumps which result in large spikes in the decoded geometry known as spiking noise. We found this problem can be filtered out by a small 3x3 median filter. Instead of directly applying the median filter, we inspect the median and if it is different than the original value, we either add or subtract 2π from the value which removes the spiking noise that is introduced by having the stair image in a lossy color channel. For this step, the unwrapped phase map is passed to the shaders and the filtered unwrapped phase map is returned.

From the filtered unwrapped phase map obtained in step (2), the normalized coordinates (x^n, y^n, z^n) can be calculated, as (Karpinsky and Zhang, 2010a)

$$x^n = j/W, \tag{7.10}$$

$$y^n = i/W, \tag{7.11}$$

$$z^n = \frac{P\Phi(x, y) - 2\pi i \cos(\theta)}{2\pi W \sin\theta}. \tag{7.12}$$

This yields a value z^n in terms of P which is the fringe pitch, i, the index of the pixel being decoded in the Holovideo frame, θ, the angle between the capture plane and the projection plane ($\theta = 30°$ for our case), and W, the number of pixels horizontally.

From the normalized coordinates (x^n, y^n, z^n), the original 3D coordinates can recovered point by point forming a floating point depth map which is step (3) in the decoding process.

$$x = x^n \times S_c + C_x, \tag{7.13}$$

$$y = y^n \times S_c + C_y, \tag{7.14}$$

$$z = z^n \times S_c + C_z. \tag{7.15}$$

Here S_c is the scaling factor to normalize the 3D geometry and (C_x, C_y, C_z) are the center coordinates of the original 3D geometry.

Now that the depth map has been calculated, step (4) normal calculation can be performed. This is done by calculating normalized surface normals with adjacent polygons on the depth map, and then normalizing the sum of these adjacent surface normals to get a normalized point normal. These values are passed out of the shader as a texture which forms the normal map.

Finally, the final rendering pass can be performed, step (5). Before this step is performed, the projection is switched back to a perspective projection, and the back screen buffer is bound as the draw buffer. Then the final render shaders are bound and a plane of pixels is rendered out. In the vertex shader, the vertex is modified according to the depth map. In the fragment shader, per pixel lighting is applied using the normal map calculated during step (4). To subsample the geometry, the number of pixels rendered out in this stage can be reduced by some divisor of the width and height of the Holovideo frame. This allows for a simple subsampling mechanism, since the points will not get calculated during the shader passes, reducing the level of detail and computational load. This is what allows the Holovideo technique to scale to different devices with various graphics capabilities.

7.4 Experimental results

In all of our experiments we used a Holovideo system configured as follows: $512(W) \times 512(H)$ image resolution, $\theta = 30°$ for the angle between the projection and capture

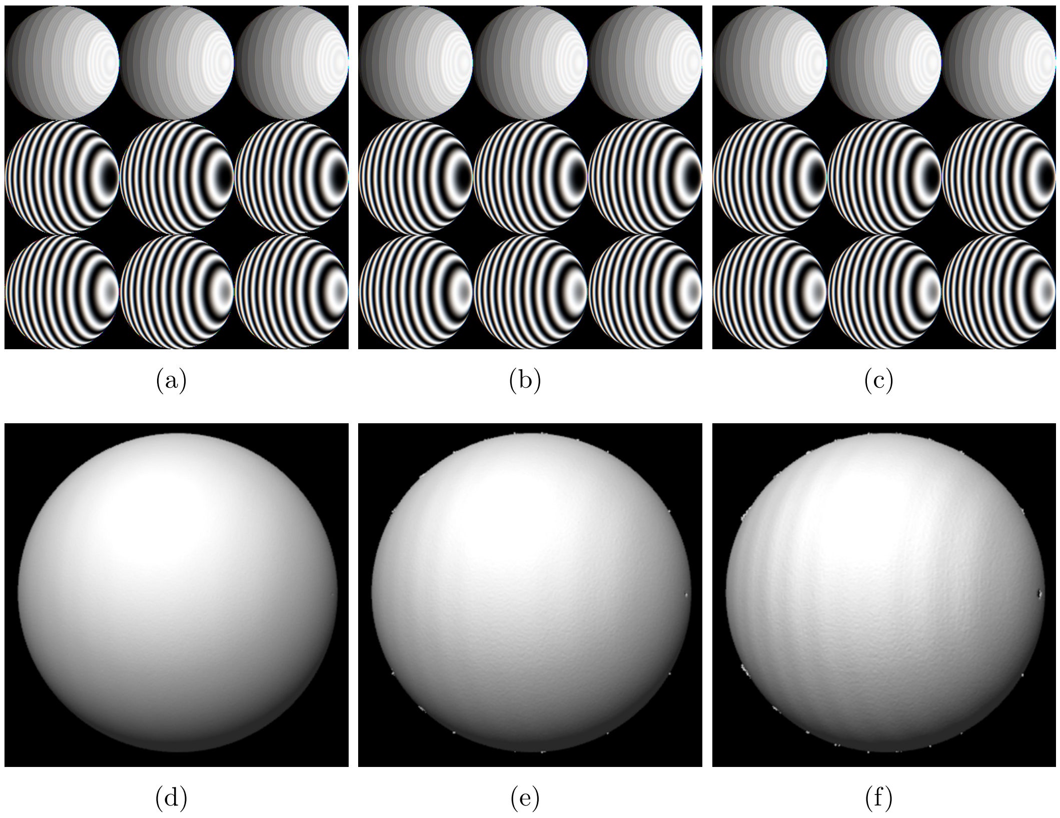


Figure 7.6 The effect of downsampling and compressing the unit sphere with the H.264 codec. (a) original planar YUV Holovideo frame, (b) recovered planar YUV444 Holovideo frame, (c) recovered planar YUV422 Holovideo frame. (d)-(f) corresponding reconstructed unit sphere.

planes, fringe pitch $P = 42$, and high-frequency modulation pitch $P = 4$. Previous work has compared the Holovideo technique to other techniques (Karpinsky and Zhang, 2012b), in this discussion we will focus on the results of applying Holovideo with H.264 encoding. To verify the performance of the proposed encoding system, we first encoded a unit sphere which represents smooth changing geometry. Figure 7.6 shows the results. Figure 7.6 (a) shows the original planar YUV Holovideo frame, Figure 7.6 (b) shows the recovered planar YUV444 Holovideo frame, and Figure 7.6 (c) shows the recovered planar YUV422 Holovideo frame. Figures 7.6 (d)-(f) show the reconstructed unit spheres with their respective encoding, Figures 7.7 (a)-(c) show cross sections of the sphere, and Figures 7.7 (d)-(f) show plots of the ΔZ. When downsampling on the color channels is

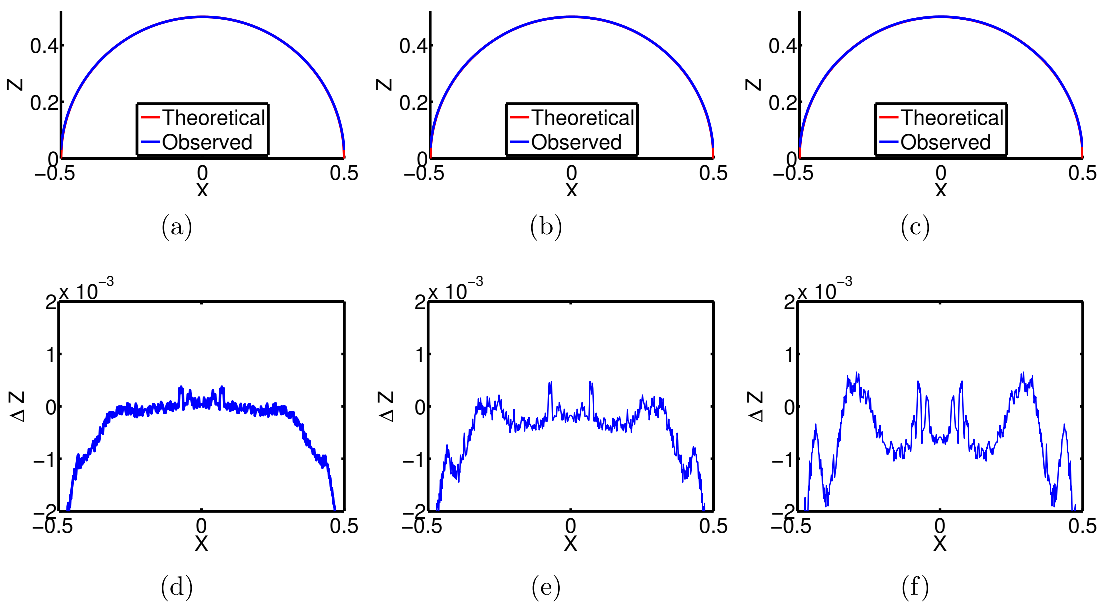


Figure 7.7 The effect of downsampling and compressing the unit sphere with the H.264 codec. (a)-(c) cross section of reconstructed unit sphere from Figure 7.6. (d)-(f) ΔZ between theoretical and observed value.

used, bands corresponding to the fringe patterns show up on the geometry as noise, but the mean squared error remains at 0.204% and 0.415% respectively.

To further test the performance of the proposed encoding system, we used it on a short 45 second 3D video clip captured using a structured light scanner with an image resolution of $640(H) \times 480(W)$ running at 30 frames per second (Zhang and Huang, 2006a). The 3D frames were Holovideo encoded and run through the H.264 encoder using the FFMPEG *lossless_max* preset in a YUV444 color format, and using the FFMPEG *baseline* preset in a YUV422 color format. Figure 7.8 (a) shows a sample frame from the scanner (associated Video 1), (b) the *lossless_max* preset Holoencoded frame (associated Video 2), and (c) the *baseline* preset Holoencoded frame (associated Video 3). The original size of the video in the OBJ format was 42 GB. In the *lossless_max* preset Holovideo format, the resulting video size is 119MB resulting in a compression ratio of over 352 : 1. Using the *baseline* preset Holovideo format, the resulting video size is 6.9 MB resulting in a compression ratio of over 6086 : 1.

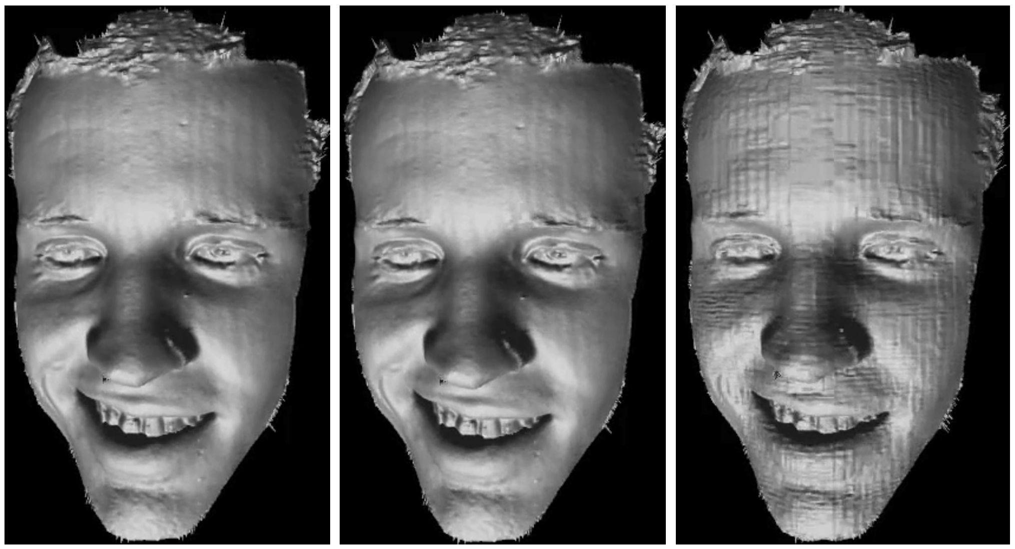


Figure 7.8 Sample frame from a short 45 second 3D video clip captured using a real-time structured light scanner running at 30 frames per second. (a) shows the original rendered data (Video 1), (b) shows the data with H.264 Holovideo encoding using the FFMPEG *lossless_max* preset (Video 2), (c) shows the data with H.264 Holovideo encoding using the FFMPEG *baseline* settings (Video 3). FFMPEG *lossless_max* settings in the YUV444 color format give a compression ratio of over 352 : 1, and FFMPEG *baseline* settings in the YUV422 color format give a compression ratio of over 6086 : 1.

7.5 Conclusion

We have presented a way to utilize the Holovideo technique with 2D video codecs that use the YUV color space, specifically the H.264 codec. Example frames of a unit sphere and a recorded data set were encoded, decoded, and presented. A mean squared error of only .204% was achieved when using the planar YUV444 color space and .415% when using planar YUV422. Applying the technique to a short 45 second 3D video clip captured using a real-time structured light scanner, a compression ratio of over 352 : 1 was achieved when compared to the OBJ file format. Currently, decoding at 28 frames per second with an NVIDIA GeForce 9400m GPU can be achieved, with encoding at 17 frames per second. Future work for this research includes ways of minimizing interframe changes to optimize the H.264 codec's keyframe and interframe coding to maximize compression, along with optimizing encoding parameters to achieve high compression with little loss of quality.

CHAPTER 8. Three bit representation of three-dimensional range data

A paper published in the Journal of Applied Optics in April 2013.[1]

Nikolaus Karpinsky[2,5], Yajun Wang[3,5], and Song Zhang[4,5]

8.1 Abstract

Our previous research has shown that three-dimensional (3D) range data sizes can be substantially reduced if they are converted into regular 2D images using the Holoimage technique. Yet, this technique requires all 24 bits of a standard image to represent one 3D point, making it impossible for a regular 2D image to carry 2D texture information as well. This paper proposes a novel approach to represent 3D range data with 3 bits, further reducing the data size. We demonstrate that over 8.2:1 compression ratio can be achieved with compression root-mean-square (rms) error of only 0.34%. Moreover, we can use another bit to represent a black-and-white 2D texture, and thus both 3D data and 2D texture images can be stored into an 8-bit grayscale image. Both simulation and experiments are presented to verify the performance of the proposed technique.

[1]This paper was published in the Journal of Applied Optics and is made available as an electronic reprint with the permission of OSA. The paper can be found at the following URL on the OSA website: http://www.opticsinfobase.org/ao/abstract.cfm?uri=ao-52-11-2286. Systematic or multiple reproduction of distribution to multiple locations via electronic or other means is prohibited and is subject to penalties under law.

[2]Primary researcher and author

[3]Graduate student collaborator

[4]Assistant professor and advisor.

[5]Department of Mechanical Engineering, Iowa State University

8.2 Introduction

Advancements in real-time 3D scanning are being made at an unprecedented rate, driving the technology further into mainstream life, as can be seen from real-time 3D scanners such as the Microsoft Kinect (Geng, 2011; Zhang, 2010). With these advancements, large amounts of data are being generated, bringing forth the challenge of streaming and storing this information in an efficient manner. Classical geometry compression approaches compress the 3D geometry and its attributes such as normals, texture coordinates, etcetera, in a model format such as OBJ, PLY, STL. Though these formats work well for static scans or structured meshes, the same does not hold true for 3D scans from a real-time 3D scanner due to its unstructured nature (Karpinsky and Zhang, 2012b).

To address this challenge newer approaches better suited to data coming from 3D scanners have been developed, including heuristic based point cloud encoding (Merry et al., 2006; Gumhold et al., 2005) and image based encoding approaches (Gu et al., 2002; Krishnamurthy et al., 2001; Gu et al., 2006). Image based encoding approaches work well as the geometry can be projected into images, then 2D image compression can be utilized until 3D reconstruction is desired. Since 2D image compression is a long studied field, high compression ratios with relatively low amounts of error can achieved.

Holoimage (Gu et al., 2006) is a image based encoding technique that has been developed, which allows for real-time encoding and decoding at high compression ratios. It leverages techniques from optical metrology, namely fringe projection. Due to the error tolerance in fringe projection, the fringe patterns can be highly compressed with little error to the reconstructed 3D geometry. Karpinsky and Zhang (Karpinsky and Zhang, 2010a) proposed to utilize the Holoimage technique and Hou et al. (Hou et al., 2012) proposed a similar virtual structured light technique to compress 3D geometry. Based on Holoimage's real-time encoding and decoding, it is able to compress data from real-time 3D scanners (Karpinsky and Zhang, 2012b). With these merits, it is well suited as a format for high speed 3D scans, which can then be streamed and stored.

Although Holoimage is a good technique for compressing 3D geometry from a real-time 3D scanner, it still uses 24 bits to represent a 3D coordinate, which in practice takes up the three standard image channels (Red, Green, and Blue). With this representation there is no room in a standard image for other information such as a texture or a normal map. This research addresses this by representing the image with only 3 bits instead of 24 through the use of image dithering. This leaves 21 remaining bits for other information such as texture or normal maps, allowing for more information to be stored and streamed. With this new encoding, compression ratios of 8.1 : 1 have been achieved when compared with a 24 bit Holoimage with a mean squared error of .34%.

Section 8.3 explains the principle behind Holoimage, applying image dithering, and how it fits into the Holoimage pipeline. Section 8.4 shows experimental results of a 3D unit sphere and David bust and discusses the findings. Finally, Section 8.5 summarizes the paper.

8.3 Principle

8.3.1 Holoimage Encoding and Decoding

Holoimage is a form a 3D geometry representation that is well suited to quickly and efficiently compressing 3D geometry coming from 3D scanners (Karpinsky and Zhang, 2010a). It works off of the principle of fringe projection from optical metrology. Encoding works by creating a virtual fringe projection system and virtually scanning 3D geometry into a set of 2D images which can then later be used to decode back into 3D. Figure 8.1 shows a conceptual model of the Holovideo system. The projector projects a pattern onto the geometry, which can be done using OpenGL shaders (Karpinsky and Zhang, 2012b), and then the camera captures the resulting scene, which can be done by saving the framebuffer as an image. Once in the image format, standard 2D image processing techniques such as compression or dithering can be applied.

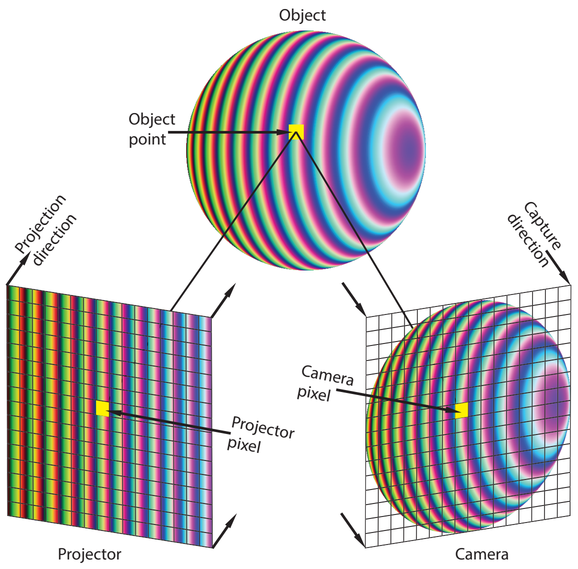


Figure 8.1 Holovideo system conceptual model. The virtual projection system projects sinusoidal fringe patterns onto the object, the result is rendered by the graphics pipeline, and then displayed on the screen. The screen view acts as a virtual camera imaging system. Because both the projector and the camera are virtually constructed, they can both be orthogonal devices. The angle between the projection system and the camera imaging system is θ.

Details of the Holoimaging encoding and decoding algorithms have been thoroughly discussed in Ref. (Karpinsky and Zhang, 2012b), we only briefly explain these algorithms here. The Holoimage encoding colors the scene with the structured light pattern. To accomplish this, the model view matrix of the projector is rotated around the z axis by some angle (e.g., $\theta = 30°$) from the camera matrix. Each point is colored with the following three equations,

$$I_r(x,y) = 0.5 + 0.5\sin(2\pi x/P), \tag{8.1}$$

$$I_g(x,y) = 0.5 + 0.5\cos(2\pi x/P), \tag{8.2}$$

$$I_b(x,y) = S \cdot Fl(x/P) + S/2 + (S-2)/2 \cdot \cos[2\pi \cdot Mod(x,P)/P_1], \tag{8.3}$$

Here P is the fringe pitch, the number of pixels per fringe stripe, $P_1 = P/(K+0.5)$ is the local fringe pitch and K is an integer number, S is the stair height in grayscale intensity value, $Mod(a, b)$ is the modulus operator to get a over b, and $Fl(x)$ is to get the integer number of x by removing the decimals.

Decoding the resulting Holoimage is more involved than encoding involving four major steps: (1) calculating the phase map from the Holoimage frame, (2) filtering the phase map, (3) calculating normals from the phase map, and (4) performing the final render. A multipass rendering was utilized to accomplish these steps, saving results from the intermediate steps to a texture, which allowed us to access neighboring pixel values in proceeding steps.

Equations (8.1)–(8.3) provide the phase uniquely for each point,

$$\Phi(x, y) = 2\pi \times Fl[(I_b - S/2)/S] + \tan^{-1}[(I_r - 0.5)/(I_g - 0.5)]. \tag{8.4}$$

It should be noted the this phase is already unwrapped, and thus no spatial phase unwrapping is required for this process. From the unwrapped phase $\Phi(x, y)$, the normalized coordinates (x^n, y^n, z^n) can be decoded as (Karpinsky and Zhang, 2010a)

$$x^n = j/W, \tag{8.5}$$

$$y^n = i/W, \tag{8.6}$$

$$z^n = \frac{P\Phi(x, y) - 2\pi i \cos(\theta)}{2\pi W \sin\theta}. \tag{8.7}$$

This yields a value z^n in terms of P which is the fringe pitch, i, the index of the pixel being decoded in the Holoimage frame, θ, the angle between the capture plane and the projection plane ($\theta = 30°$ for our case), and W, the number of pixels horizontally.

From the normalized coordinates (x^n, y^n, z^n), the original 3D coordinates can recovered point by point

$$x = x^n \times S_c + C_x, \tag{8.8}$$

$$y = y^n \times S_c + C_y, \tag{8.9}$$

$$z = z^n \times S_c + C_z. \tag{8.10}$$

Here S_c is the scaling factor to normalize the 3D geometry, (C_x, C_y, C_z) are the center coordinates of the original 3D geometry.

8.3.2 Image Dithering

Image dithering is the process of taking a higher color depth image and reducing the color depth to a lower level through a quantization technique (Schuchman, 1964). Different types of image dithering techniques exist such as ordered dithering (Bayer, 1973) and error diffusing (Kite et al., 2000). In this research, two of the most popular algorithms were investigated, Bayer (Bayer, 1973) and Floyd-Steinberg (Floyd, 1976) dithering.

8.3.2.1 Bayer Dithering

Bayer dithering, sometimes known as ordered dithering, involves quantizing pixels based on a threshold matrix (Bayer, 1973). In the simple case of quantizing to a binary image, it involves taking each pixel in an image and applying Algorithm 1.

```
Input: Pixel - Structure representing properties of a pixel in an image. Has color
       components ranging from 0.0 to 1.0
Input: ThresholdMap - Matrix of threshold values
Output: Pixel.color - Pixel's dithered color component, either 0 or 1
for Each Pixel do
    if Pixel.color >= ThresholdMap[pixel.x mod mapWidth][pixel.y mod
    mapHeight] then
        Pixel.color = 1;
    else
        Pixel.color = 0;
    end
end
```

Algorithm 1: Bayer Dithering

$$M = \frac{4.0}{255.0} * \begin{bmatrix} 0 & 32 & 8 & 40 & 2 & 34 & 10 & 42 \\ 48 & 16 & 56 & 24 & 50 & 18 & 58 & 26 \\ 12 & 44 & 4 & 36 & 14 & 46 & 6 & 38 \\ 60 & 28 & 52 & 20 & 62 & 30 & 54 & 22 \\ 3 & 35 & 11 & 43 & 1 & 33 & 9 & 41 \\ 51 & 19 & 59 & 27 & 49 & 17 & 57 & 25 \\ 15 & 47 & 7 & 39 & 13 & 45 & 5 & 37 \\ 63 & 31 & 55 & 23 & 61 & 29 & 53 & 21 \end{bmatrix} \tag{8.11}$$

Equation (8.11) gives an example of an 8 × 8 threshold matrix, which was also the matrix used in this work.

Bayer has shown that if the sizes of the matrices are 2^N (N is an integer), then optimal matrices can be derived; the matrices can be obtained as follows,

$$M_1 = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}, \tag{8.12}$$

which is the smallest 2 × 2 base dither pattern. Larger dither patterns can be obtained using

$$M_{n+1} = \begin{bmatrix} 4M_n & 4M_n + 2U_n \\ 4M_n + 3U_n & 4M_n + U_n \end{bmatrix}, \tag{8.13}$$

where U_n is an n-dimensional unit matrix (one for all elements). Using larger threshold matrices allows more distinct tones to be represented in the final image, thus larger threshold matrices could result in lower error. In this research, we used an 8 × 8 since it is a large matrix that should theoretically yield 64 different tones in the resulting image. However, we also found that if the matrix is too large, the resultant dithered pattern is not of high-quality. We typically use 8 × 8 or 16 × 16 matrices of our study.

With Bayer dithering, the threshold map adds minor local error noise to the quantized pixel, but the overall intensity is preserved. Since this algorithm is a parallel algorithm, it can easily be integrated into the Holoimage pipeline in the fragment shading stage of the encoding allowing for little to no overhead in encoding.

8.3.2.2 Floyd-Steinberg Dithering

Floyd-Steinberg dithering is a form of error diffusing dither, which diffuses quantization error of a specific pixel into neighboring pixels(Floyd, 1976). Through error diffusing, the cumulative quantization error is kept to a minimum, which is near zero. The original Floyd-Steinberg dithering algorithm is given with Algorithm 2.

```
Input: Image - Original Image to be dithered. Has color components ranging
       from 0.0 to 1.0
for $y \leftarrow 0$ to $Image.Height$ do
    for $x \leftarrow 0$ to $Image.Width$ do
        if $Image.Pixel(x,y).color >= 0.5$ then
            newColor = 1;
        else
            newColor = 0;
        end
        quantError = Image.Pixel(x,y) - newColor;
        Image.Pixel(x,y) = newColor;
         //Diffuse Error;
        Image.Pixel(x+1,y) += 7/16 * quantError;
        Image.Pixel(x-1,y+1) += 3/16 * quantError;
        Image.Pixel(x,y+1) += 5/16 * quantError;
        Image.Pixel(x+1,y+1) += 1/16 * quantError;
    end
end
```

Algorithm 2: Floyd-Steinberg Dithering

In the first part of the algorithm, the images pixel value is quantized into either 1 or 0. Then the quantization error from this operation is calculated, and then diffused into neighboring pixels, to the right and down. It should be noted that unlike ordered dithering, this algorithm is a serial algorithm, operating on the image pixels one by one. In the standard Floyd-Steinberg dithering algorithm, the route is from left to right, top to bottom, realized as a forward array. Once a pixel has been quantized it is no longer changed. However, we did find that the resultant image quality depends on the starting location and path of diffusing the error (Lohry and Zhang, 2013). Another zigzag route could be taken, but the algorithm would have to be altered slightly so that the error is not diffused into pixels that have been dithered in the new zigzag route.

8.4 Experiments

To test the effects of image dithering on Holoimages, we performed both Bayer and Floyd-Steinberg dithering on Holoimages of a unit sphere and 3D scan of the statue of David. In all of our experiments we had a fringe frequency of 12, θ of 30 deg, and Holoimage size of 512×512.

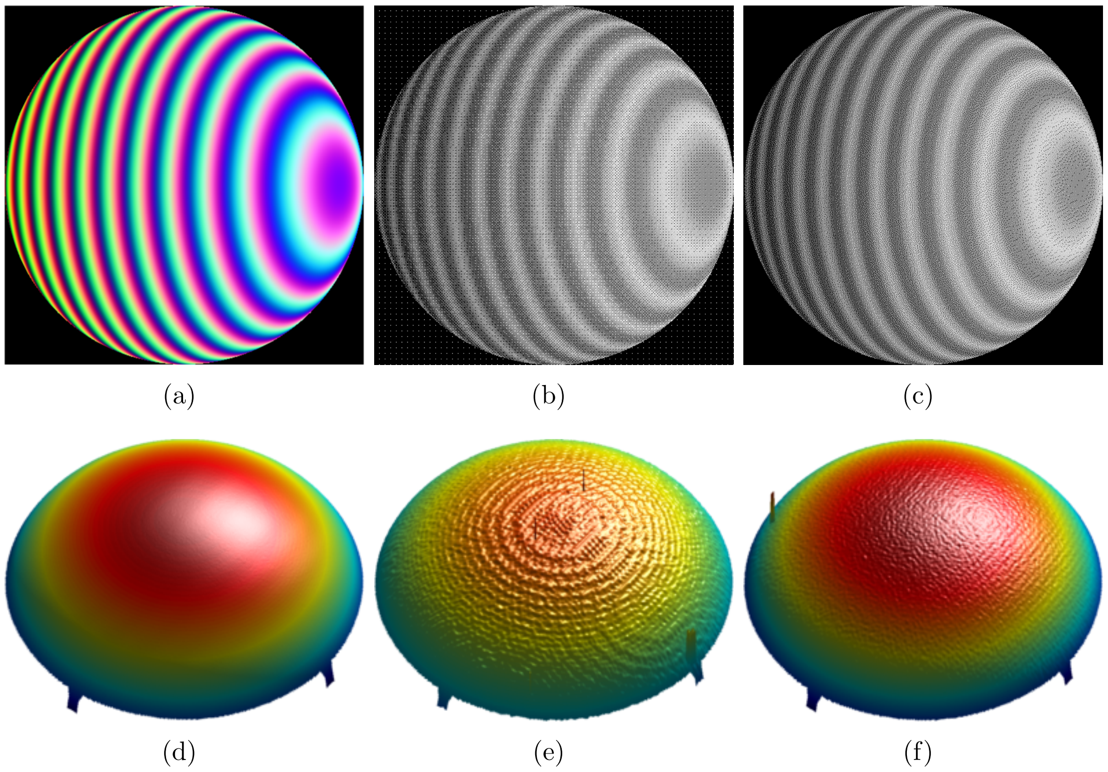


Figure 8.2 Results of dithering on unit sphere in a lossless image format. (a) Original Holoimage; (b) Holoimage with Bayer dithering; (c) Holoimage with Floyd-Steinberg dithering; (d) 3D reconstructed results for image shown in (a); (e) 3D reconstructed results for image shown in (b); (f) 3D reconstructed results for image shown in (c).

To begin we performed the dithering on the unit sphere and then stored the resulting images in the lossless PNG format. Figures 8.2 and 8.3 show the results. Figure 8.2(a) shows the Holoimage. Red, green, and blue (RGB) channels of the Holoimage are then dithered with the Bayer Dithering technique individually; and then stored into three most significant bits of the 8-bit grayscale image shown in Fig. 8.2(b) with R being stored as the most significant bit, and B being stored as the third significant bit. This grayscale image contain all the information required to recover the whole 3D geometry carried on by the 24-bit Holoimage shown in Fig. 8.2(a). Similarly, the other dithering technique can also be employed to convert the 24-bit Holoimage into the three most significant bits of an 8-bit grayscale image. Figure 8.2(c) shows the dithered image using the Floyd-Steinberg dithering technique.

Before the 3D geometry can be decoded from the Holoimage, 2D image processing needs to be reversed to attempt to put the Holoimage back into its original state. In terms of dithering, this can be done by applying a low-pass filter, such as a Gaussian filter, to the dithered image. In this research, we used a 7×7 Gaussian filter with a standard deviation of $7/3$ pixels. It is also important to know that in the Holoimage pipeline, filtering can be applied after phase unwrapping. Previous work has shown that median filtering can remove spiking noise in the final reconstruction (Karpinsky and Zhang, 2012a; McGuire, 2008). This is done by median filtering, and then instead of using the median, detecting the correct number of phase jumps from the median and applying it to the phase at the current pixel.

Figure 8.2(e) shows the reconstructed result from the Bayer-dithered pattern shown in Fig. 8.2(b). In comparison with the 3D result recovered from 24-bit Holoimage shown in Fig. 8.2(d), the Bayer-dithered result has some random noise on top of the recovered 3D results. Yet, the sphere was well recovered. Figure 8.2(f) shows the recovered results using the Floyd-Steinberg dithering technique, it is significantly better than the result obtained from the Bayer-dithering technique.

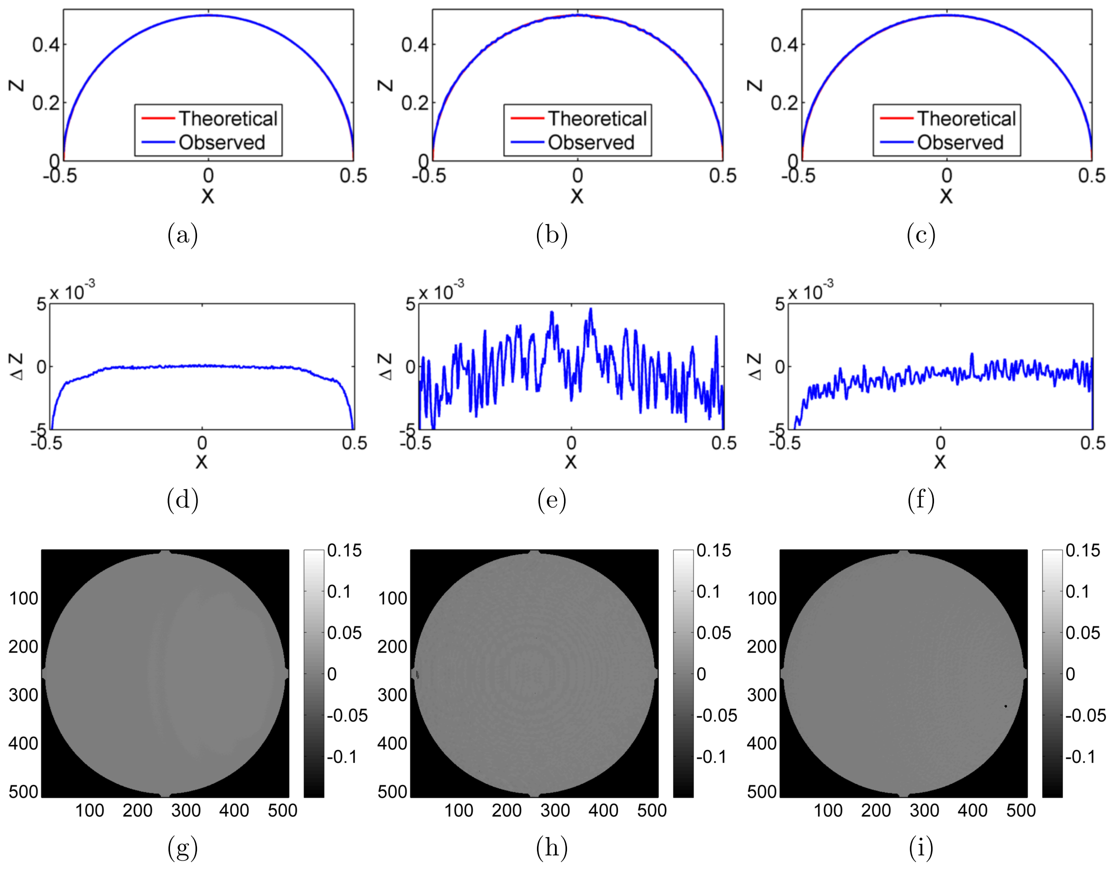


Figure 8.3 Reconstruction errors of dithering on unit sphere in a lossless image format. (a) Cross section of reconstructed result show in Fig. 8.2(d); (b) Cross section of reconstructed result show in Fig. 8.2(e); (c) Cross section of reconstructed result show in Fig. 8.2(f); (d) Reconstruction error between the reconstructed and ideal unit sphere for the result in (a); (e) Reconstruction error between the reconstructed and ideal unit sphere for the result in (b) (approximately rms error 0.33%); (d) Reconstruction error between the reconstructed and ideal unit sphere for the result in (c) (approximately rms error 0.2%); (g-i) difference map of technique to ideal unit sphere.

To better compare these dithering techniques, Figure 8.3 shows the cross sections of the recovered 3D results using different methods comparing with the ideal unit sphere. Figures 8.3(a) and 8.3(d), respectively, show that the cross section of recovered 3D sphere overlapping with the ideal unit sphere, and the cross section of the difference between these two, when the 24-bit Holoimage is used. The results are smooth, and the error is small, which has been demonstrated previously (Karpinsky and Zhang, 2012b). The Bayer-dithered results (Fig. 8.3(b) and Fig. 8.3(e)) shows that the overall geometry was recovered quite well, but error is larger: approximately root-mean-square (rms) error of 0.33%. It can be seen that this error is still quite small. Yet, the Floyd-Steinberg dithering technique can further improve the results, as shown in Fig. 8.3(c) and Fig. 8.3(f). This is due to the quantization error being diffused into neighboring pixels, reducing the overall quantization error. The error can be further reduced to be approximately rms 0.2%. Is should be noted that only 3 bits were used to represent the 24 bit Holoimage, and the reconstructed geometry is still high quality.

Compression results depend on how the resulting dithered information is stored. In this work JPEG and other lossy image compression was not used due to the fact that it makes use of a low pass filter before compression. This takes the 3 bit binary dithered information and transforms it back into 24 bit information, which is undesirable. Instead, PNG a lossless image compression, was utilized and the three most significant bits of a grayscale image were utilized, show by Fig. 8.4 (a). This resulted in a file size of 79 KB with the unit sphere. Further compression can be achieved by saving the image in a planar format, three times as wide with image channels one after another, and then saving the PNG as a logical 1 bit image. This resulted in a file size of 62 KB yielding a compression ratio of 3.9 : 1 when compared against a 24 bit Holoimage in the PNG format.

To further test dithering on Holoimages, the technique was performed on a scan of the statue of David shown in Fig. 8.5. Figure 8.5(a) shows the 24-bit Holoimage,

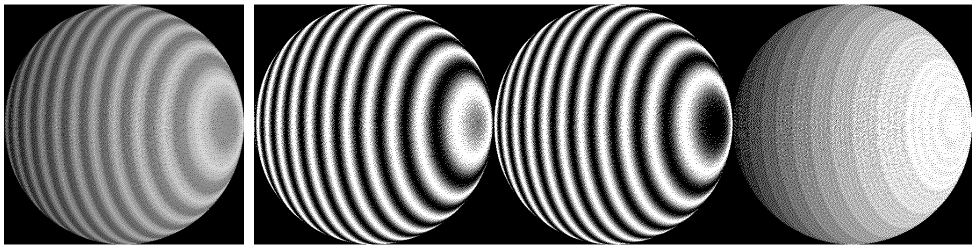


Figure 8.4 Different ways to hold a packed dithered Holoimage. (a) Dithered channels packed in three most significant bits and saved as grayscale PNG with resulting file size of 79 KB; (b) Dithered channels packed into a planar format and then saved as a logical PNG with resulting filesize of 62 KB.

and Figure 8.5(d) shows the recovered 3D geometry. The 24-bit Holoimage is then dithered into 3-bits and stored into three most significant bits of a 8-bit grayscale image. Figure 8.5(b) and Fig. 8.5(c), respectively, shows the Bayer-dithered result and the Floyd-Steinberg dithered result; and their recovered 3D shapes are shown in Fig. 8.5(e) and Fig. 8.5(f). Again, it can be seen that Bayer dithering results in larger amounts of error seen as ripples and bumps on the surface; and Floyd-Steinberg dithering has some of these errors as well but it is not as prominent as is the case with Bayer dithering. Floyd-Steinberg dithering results a lower rms error or .34% when compared to Bayer dithering at .37%. The resulting file size is 39 KB, achieving a compression ratio of 8.2 : 1 when compared to the 24 bit Holoimage. Although it might be expected a simple unit sphere would have a higher compression, this is not the case as PNG compression depends on pre-compression and DEFLATE steps which can result in different file sizes for similar images.

Since the proposed technique only requires 3 bits to represent the whole 3D geometry, there are 21 bits remaining to encode more information such as the grayscale texture that comes from the 3D scanner, which can be encoded into the same image. There are essentially two approaches to carry on texture with 3D geometry. The first method is to pack the 8-bit grayscale image directly into the 24-bit image. Figure 8.6(a) shows

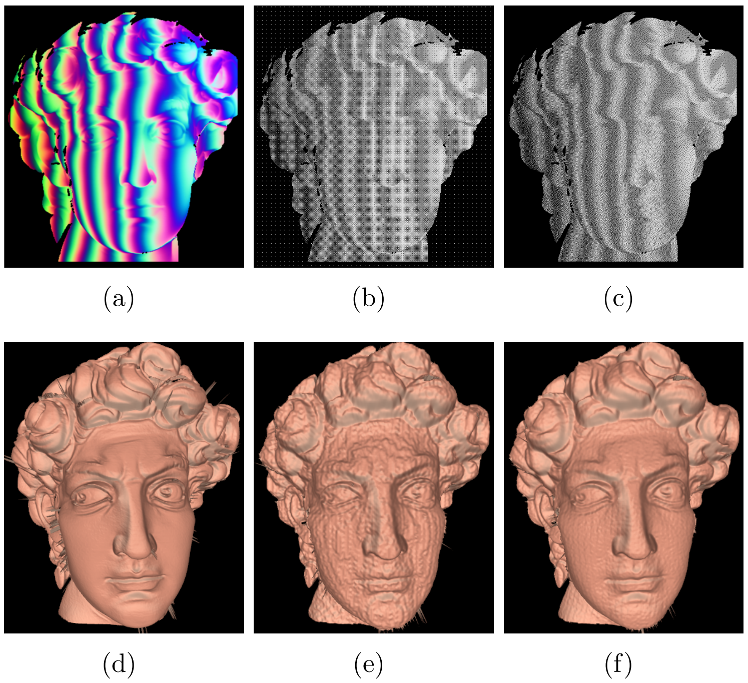


Figure 8.5 Results of dithering on scan of David statue in a lossless image format. (a) original Holoimage; (b) Holoimage with Bayer dithering; (c) Holoimage with Floyd-Steinberg dithering; (d) Recovered 3D geometry from (a); (e) Recovered 3D geometry from (b); (f) Recovered 3D geometry from (c).

the resultant image, and its recovered 3D geometry with texture mapping is shown in Fig. 8.6(b). The file size is approximately 189 KB, which is a substantial reduction comparing with original 24-bit Holoimage stored in PNG format 320 KB.

The 8-bit texture image can be dithered as well to further compress the data. Figure 8.6(c) shows the packed image that stores the 3D geometry along with 1-bit dithered texture image into four most significant bits of a 8-bit grayscale image. From this image, the texture can be recovered by applying a very small Gaussian filter (7×7) as shown in Fig. 8.6(d). It can be seen that the texture image is of good quality. With only 4 bits, the file size is approximately 64 KB. This example clearly demonstrates that the proposed technique can embed both the 3D geometry and the texture into a regular 2D image, making it a novel technique to store 3D range data in a substantially reduced size, with minor loss of quality. Furthermore, because it only utilizes 4 bits, this proposed 3D

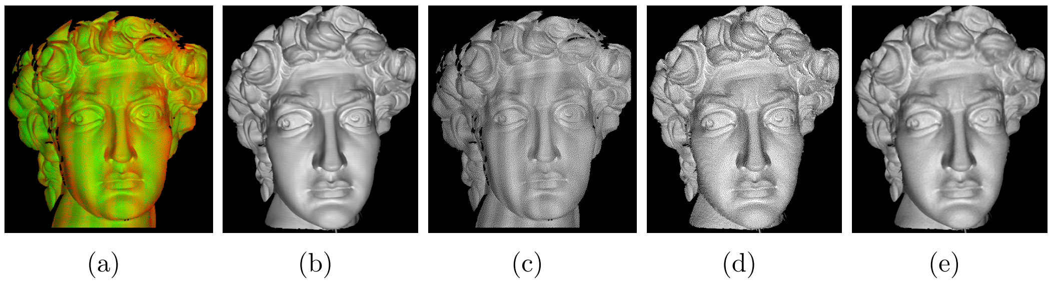


Figure 8.6 Packing dithered Holoimage with texture. (a) 3-bit packed Holoimage with 8-bit grayscale texture; (b) 3D geometry with original 8-bit texture mapping; (c) 3-bit packed Holoimage with 1-bit dithered texture; (d) 3D geometry with 1-bit dithered texture mapping; (e) 3D geometry with 1-bit dithered texture after texture is Gaussian filtered.

range data compression technique can be leveraged for applications where other critical information such as connectivity or bump maps need to be carried along.

8.5 Conclusion

A novel approach to represent 3D geometry has been presented, specifically applying image dithering to the Holoimage technique to reduce the bit depth from 24 bits to 3 bits. The technique was presented with two forms of image dithering, and sample data of a unit sphere and 3D scan of David have been demonstrated. A mean squared error of .2% was achieved on the unit sphere with a compression of 3.9 : 1 when compared with the 24 bit Holoimage technique, and a rms error of .34% was achieved on the scan of David with a compression of 8.2 : 1 when compared with the 24 bit Holoimage. With the remaining 21 bits, grayscale texture information was also encoded, effectively embedding 3D geometry and texture into a single 8-bit grayscale image. Future work for this research includes extending it to video with animation appropriate dithering.

CHAPTER 9. Portal-s: Achieving eye gaze in 3D video conferencing

A paper to be submitted to the ACM *SIGGRAPH* conference.

Nikolaus Karpinsky[1,4], James Oliver[2,4], and Song Zhang[3,4]

9.1 Abstract

To overcome the challenges associated with achieving the goal of telepresence, the portal system (Portal-s) is proposed. Portal-s consists of a set of algorithms with associated capture and display hardware, that effectively captures a user in 3D without encumbering them with additional hardware such as fiducial markers or shutter glasses, and display on another Portal-s node with correct eye gaze to achieve eye contact between users. The

[1]Primary researcher and author
[2]Professor and co-advisor
[3]Assistant professor and advisor.
[4]Department of Mechanical Engineering, Iowa State University

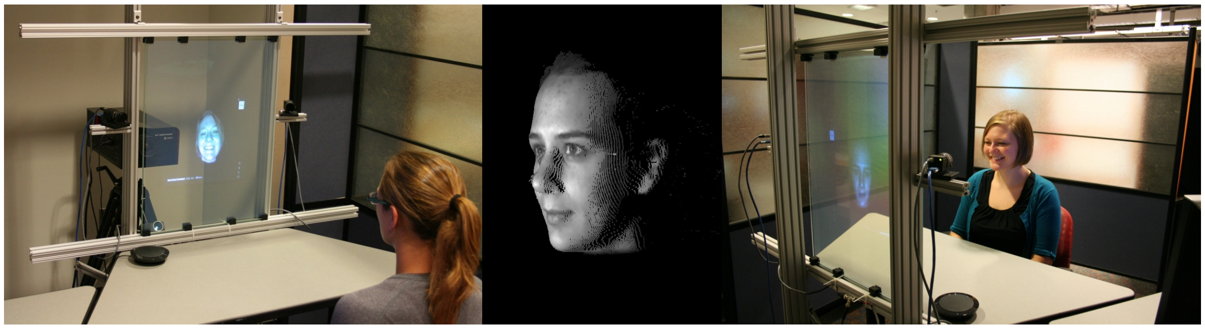


Figure 9.1 Example of a Portal-s node. User can walk up to the node, be scanned and transmitted to another connected node. Looking through the display, user sees the 3D scan from the other connected node.

3D capture system is realized as dual camera structured light scanners in the infrared (IR) spectrum, so as not to bother the users' eyes with white light used in conventional structured light scanning. Parallel calculation allows the entire processing pipeline to be realized on a graphics processing unit (GPU), thus allowing real-time speeds with high resolution. Holovideo compression addresses challenges related to transmission, namely achieving real-time speeds across standard network infrastructure without reducing the resolution of the 3D reconstruction. Finally, display is performed on a holographic display through the same optical axis as capture, correcting eye gaze and achieving eye contact.

9.2 Introduction

The goal of telepresence is to allow a person to feel as if they are present in a location other than their true location; this is typically achieved through the use of technologies that can capture, transmit, and reconstruct senses such as hearing, vision, etc (Minsky, 1980). Due to the complexity of such a goal, there are many challenges associated with the task. Looking back, telepresence can be linked to systems such as the telegraph in which language encoded through electrical pulses on a telegraph line were captured and reproduced at a remote end. As technology continued to evolve, the burdens on users of learning new languages were removed, as sensors were able to capture natural human language. Current technology has advanced significantly, where 2D video of a user can be captured, transmitted, and reconstructed, with minimal effort on the user. While significant, this technology is not without problems, one of the main drawbacks is loss of eye contact between users. Due to a disparity between the capture and redisplay points, users are unable to achieve mutual eye contact with each other, and since the signal is 2D it is difficult to transform the data into a corrected view.

With new techniques in real-time 3D scanning and compression advancing, this work aims to build and improve upon the previous research in the field. A set of algorithms and associated capture and display hardware is proposed, known as the portal system (Portal-s). Portal-s makes use of dual camera structured light systems to capture a user in 3D with high resolution. 3D video compression technology is leveraged, to allow the high resolution 3D geometry to be transmitted across standard networks without reducing resolution. Finally, redisplay is achieved on a holographic display at a remote node. Through the use of a fringe projection based technique, a Portal-s node is able to capture and transmit 3D information from the same point as its display. This allows a user to look into one Portal-s node, and see out another. Through Portal-s, we make the following contributions:

1. Real-time 3D scanning and data merging of dual camera structured light scanners on a single desktop GPU.

2. Transmission and redisplay of full resolution 3D scans across a standard 10/100 Mb network and through a local residential Internet service provider.

3. Redisplay on a remote node, providing correct eye gaze.

9.3 Background and Related Work

With the ubiquitousness of real-time 3D scanners becoming more prevalent in our daily lives, the idea of utilizing such a scanner for telepresence activities is becoming a reality. Previous works have utilized 3D scanners to create telepresence nodes, such as blue-c, and the one-to-many eye contact system (Gross et al., 2003; Jones et al., 2009). These works have advanced the state of the art, and addressed some of the challenges in creating such a system, such as capture, calibration, and redisplay.

Challenges in capture stem from the need to capture and process 3D geometry in real-time. Different techniques exist such as the visual hull approach (Matusik et al.,

2000) utilized in blue-c, structured light scanning (Zhang and Huang, 2006a) utilized in one-to-many and office of the future (Raskar et al., 1998), and stereo imaging based approaches. Designing stereo vision based systems is challenging, due to the need to find correspondences in real time. This problem is further exacerbated when users of a system have good complexion, that is void of easily discernible feature points. Structured light based systems provide a better resolution, but conventional systems utilize white light based techniques which becomes a nuisance for users being scanned.

Challenges in calibration and redisplay stem from the need to redisplay to a user without inhibiting them while simultaneously establishing correct eye gaze. Conventional approaches to establishing eye gaze involve either merging the capture and redisplay axis through use of beam splitters or reducing the angle of disparity until it is unnoticeable by the user. Beam splitter approaches can be tracked back to an Autocue commonly known as a teleprompter in the entertainment industry. As users read from the display, they are simultaneously captured through the display, establishing correct eye gaze (Grayson and Monk, 2003). This technique was later applied to telecommunication through the creation of the Interrotron for use in interviews (Morris, 2004). The second approach involves minimizing the distance between capture and display, while increasing the distance to the user. As the users' distance from the system increases, the angle disparity between the capture and display axis to the user deceases; research has shown that users perceive correct eye gaze at an angle of less than 5° (Chen, 2002). This is employed in commercial systems such as the Cisco telepresence end-points, giving users the perception of correct mutual eye gaze.

9.4 Portal-s System Overview

The proposed system, Portal-s, is broken down into three major modules: capture, transmission, and display. To achieve real-time 3D telecommunication, real-time processing must be met across all three of Portal-s modules. To accomplish this, special consideration is made for both hardware and software architecture. The following sections will dissect the software and hardware architecture giving implementation details for each module.

9.4.1 Capture

The Portal-s capture module consists of three major subsystems: Capture, Processing, and Streaming. All three subsystems require a significant processing pipeline to achieve real-time speeds, with the capture subsystem also requiring significant hardware architecture. Due to the requirement of being real-time, each subsystem runs on a separate thread, with each thread sharing OpenGL contexts; this allows each subsystem to share data through OpenGL textures between threads. To deal with data synchronization between subsystems, triple buffering is employed thus allowing a thread to read or write data without waiting on another thread (Möller et al., 2002). Once data has been written to or read from the threads active buffer, the threads' pointer is atomically switched with the third working buffer. This allows for a lock-less design and the reduction of thread contention. The remainder of this section will examine in detail the proposed software and hardware architecture of each subsystem.

Capture of the user in Portal-s is achieved through the use of dual infrared (IR) structured light systems. Figure 9.2 shows the conceptual schematic diagram of the system with the dual cameras. IR structured light is emitted from an IR projector through the holographic display and onto the user. Due to the properties of the holographic display, light projected perpendicular to the display itself passes through with 98% transmittance (Vislogix, 2013). From here each camera on the sides of the display capture the resulting scene with the structured light on the user. The cameras view the scene from outside of the holographic display so that they do not observe the small amounts of IR light that is reflected back by the display. It should be noted that the dual cameras are not used in a stereo configuration, rather they are used to increase the field of view (FOV) of the 3D scanning system (Weise et al., 2007); this results in two separate structured light scans that must be later merged into a single scene.

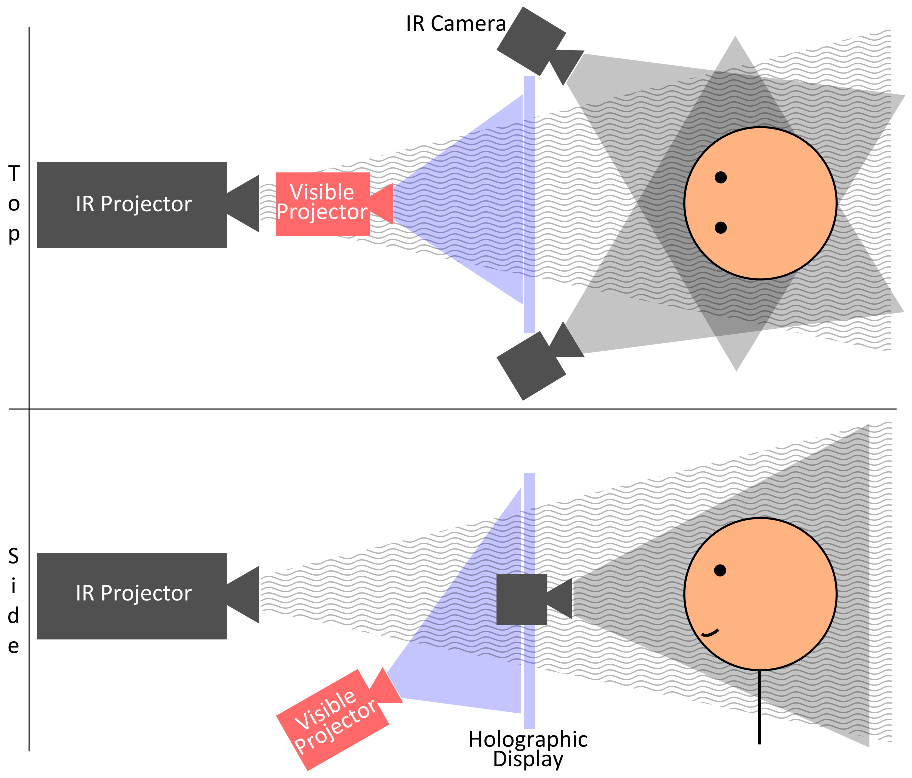


Figure 9.2 Conceptual schematic of Portal-s hardware. The top shows a top down view with the user in front of a Portal-s node, the bottom the corresponding side view. IR structured light is emitted from the IR projector, through the holographic display, and onto the user. The two IR cameras mounted on each side of the display capture the resulting scene and are used for 3D reconstruction. The visible light projector along with the holographic display form the display for the user to see the 3D scan from another Portal-s node.

The chosen structured light patterns are two frequencies of three-step phase-shifted patterns, known as the six fringe technique (Wang and Zhang, 2011). The motivation behind using six patterns instead of the commonly used three, is that with multiple frequencies, the phase can be unwrapped in parallel; conventional three-step phase-shifting algorithms require spatial phase unwrapping which is a serial process that is ill-suited to GPU implementation. Defocused binary dithering is used in the pattern projection to relax camera projector exposure timing requirements (Li et al., 2010), and remove the need of projector nonlinearity gamma correction (Lei and Zhang, 2010).

Once both cameras are synchronized together with the projector, and frames are streamed to the capture computer, the capture software can begin processing. Due to the patterns forming a set, six images in total, images cannot be dropped if the capture processing cannot keep up. To accommodate for this, each camera is run on its own thread on the capture computer. If the software detects a dropped frame, the appropriate number of frames are dropped to ensure the order is maintained. The basic image capture event loop simply grabs a grayscale frame from the camera, packs the image into one channel of a three-channel image, then transfers it to an OpenGL texture via a direct memory access (DMA) buffer transfer. Each three-channel image represents the three-step phase-shifted pattern for a specific frequency. As previously stated, each texture transfer to or from the GPU is triple buffered to allow simultaneous reads and writes, with a simple lock-less synchronization design.

Processing of the data is performed entirely on the GPU to achieve real-time speeds. Once the fringe images from the cameras are packed and streamed to the GPU as textures, processing them into a Holovideo encoded frame is achieved through multipass rendering (Karpinsky and Zhang, 2012b). The stages in the multipass rendering pipeline are: phase wrapping, phase filtering, phase unwrapping, phase filtering, coordinate calculation, merging, and Holovideo encoding. The processing pipeline is illustrated with Figure 9.3. The first five stages in this pipeline are done with OpenGL Shading Language (GLSL) image processing and performed for each camera. We use the standard technique of rendering out a screen aligned quad to an offscreen frame buffer object (FBO) that is the size of the image with a pass through vertex shader; this results in the fragment shader being called on every pixel of the texture, yielding parallel image processing for each pixel.

As can be seen from Figure 9.3 after the camera loads the fringe into a texture the first

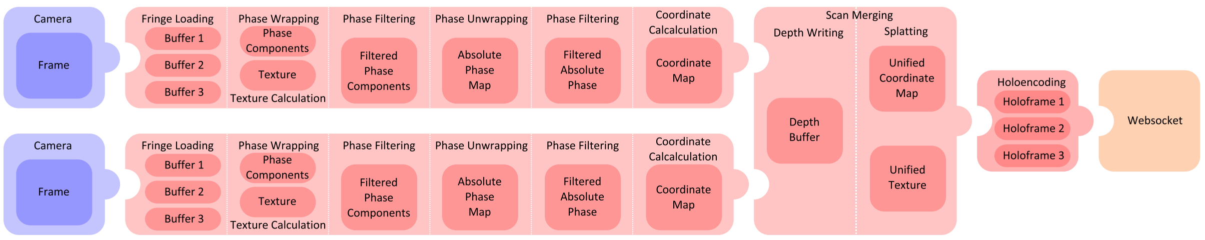


Figure 9.3 Capture processing pipeline. Blue signifies the capture thread, red the processing thread that is realized on the GPU, and finally orange signifies the streaming thread.

operation is phase wrapping and texture calculation. As previously stated, we utilized two different frequencies of three-step phase-shifted patterns which can be described as

$$I_r = I' + I''cos\left[\phi - \frac{2\pi}{3}\right], \tag{9.1}$$

$$I_g = I' + I''cos\left[\phi\right], \tag{9.2}$$

$$I_b = I' + I''cos\left[\phi + \frac{2\pi}{3}\right], \tag{9.3}$$

$$\tag{9.4}$$

where I' is the average intensity, I'' is the intensity modulation, and ϕ is the encoded phase. Texture calculation simply involves averaging one set of three phase shifted patterns together; this eliminates the fringe patterns, leaving the ambient intensity I' which is the texture

$$I' = \frac{I_r + I_g + I_b}{3}. \tag{9.5}$$

Phase wrapping is performed by applying three-step phase-shifting algorithm to each frequency of the three phase-shifted patterns, yielding a wrapped phase ϕ for each frequency.

$$\phi = \tan^{-1}\left[\frac{\sqrt{3}(I_r - I_b)}{2I_g - I_r - I_b}\right]. \tag{9.6}$$

From here, the sine and cosine components are extracted and rendered to the RGBA color channels of the output buffer

$$\begin{aligned} R &= \sin\phi_1, & (9.7)\\ G &= \cos\phi_1, & (9.8)\\ B &= \sin\phi_2, & (9.9)\\ A &= \cos\phi_2. & (9.10) \end{aligned}$$

The sine and cosine components are rendered out instead of the wrapped phase, so that during phase filtering, the phase jumps are kept intact. Next in the pipeline is phase filtering, which consists of a separable 7×7 Gaussian filter. This is done before phase unwrapping to smooth and reduce the random noise in the wrapped phase map, as small amounts of noise could pose problems for parallel phase unwrapping. After filtering, phase unwrapping is performed to recover the absolute phase Φ.

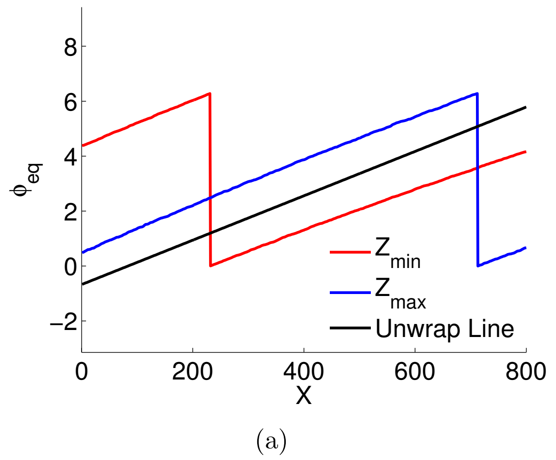
First the phase components are wrapped together again to form wrapped phase maps for each frequency

$$\begin{aligned} \phi_1 &= \tan^{-1}\left[\frac{I_r}{I_g}\right], & (9.11)\\ \phi_2 &= \tan^{-1}\left[\frac{I_b}{I_a}\right]. & (9.12) \end{aligned}$$

Next an equivalent frequency is calculated using

$$\Phi_{12} = \phi_1 - \phi_2 \mod 2\pi. \tag{9.13}$$

Assuming the frequencies for ϕ_1 and ϕ_2 are properly chosen, the equivalent frequency will span the entire image. Unfortunately, this involves choosing larger frequencies that are close together, both of which result in increased noise. To mitigate this problem, we chose frequencies that result in a single phase jump. Capturing the equivalent phase map at Z_{min} and Z_{max} the minimum and maximum depth values respectively, and plotting



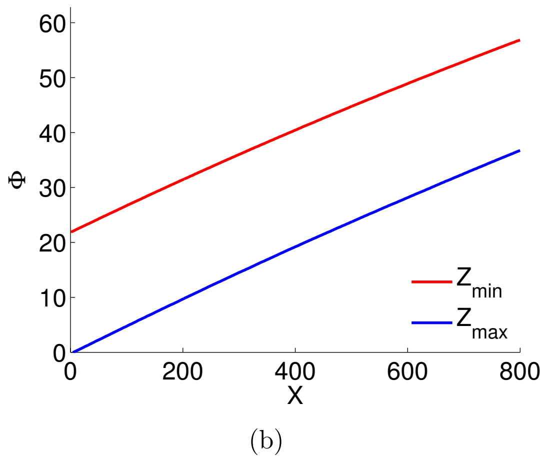


Figure 9.4 Cross section of captured phase before and after phase unwrapping. (a) cross section of wrapped phase ϕ_{eq}, red denotes the phase at Z_{min}, blue the phase at Z_{max}, and the black line is the phase unwrapping line; (b) cross section of unwrapped phase after unwrapping ϕ_{eq} with the phase unwrapping line, and using Φ_{eq} to unwrap ϕ_1.

a cross section yields Figure 9.4(a). It can be seen from the cross section that there is a region between the min and max phase that can best be separated by a line. By calibrating and saving the equation for this line, values below the line can have 2π added to them to correct for the phase jump, and values above do not. The resulting unwrapping is shown with Figure 9.4(b). After unwrapping the equivalent frequency, it can be used to unwrap ϕ_1 (denser fringe pattern) by

$$\Phi = \phi_1 + 2\pi * \text{round}\left(\frac{\phi_{12} * \frac{\lambda_{12}}{\lambda_1} - \phi_1}{2\pi}\right), \tag{9.14}$$

which is then rendered out resulting as an absolute phase map. Next in the pipeline is another round of phase filtering. This filtering aims to remove spiky noise that can be introduced during incorrect phase unwrapping. Currently, filtering is realized as a pass of an adapted 1×5 median filter, in the direction of the phase gradient, that finds the median and adjusts by the number of phase jumps (McGuire, 2008; Karpinsky and Zhang, 2012b).

The last pass in the per camera processing is coordinate calculation. Utilizing a camera projector stereo calibration (Zhang, 2000), the intrinsics and extrinsics along with the absolute phase map are used in a linear triangulation to form a coordinate map (Hartley and Zisserman, 2000; Zhang et al., 2006). The resulting coordinate map is rendered out to a floating point coordinate map. Figure 9.5 shows an example of the data from a single camera as it goes through the render passes in the pipeline.

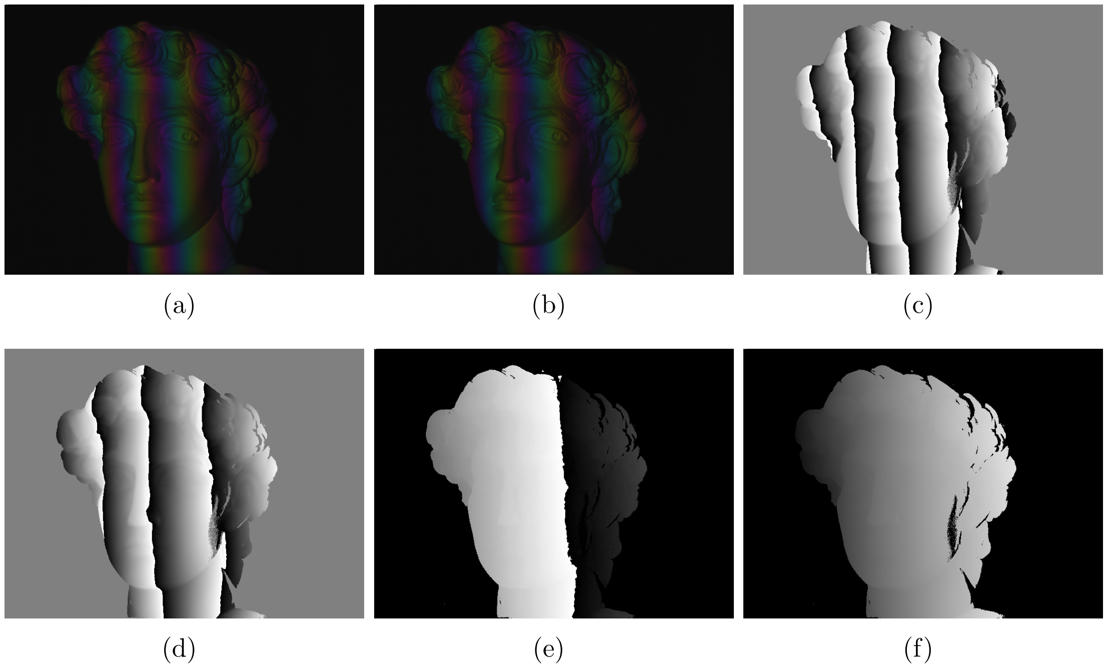


Figure 9.5 Example of six fringe structured light scanning from one of the cameras. (a)-(b) show structured light patterns projected onto the subject and captured with a frequency λ of 57 and 69 pixels respectively. (c)-(d) shows wrapped phase obtained by applying Equation 9.6 to each set of patterns. (e) shows the wrapped equivalent phase, (f) shows the resulting unwrapped phase.

At this point the two camera views need to be merged into a single view. Normally, this would involve using an alignment algorithm such as ICP which works great for stationary objects (Rusinkiewicz and Levoy, 2001). However our object is dynamically changing and it is difficult to adopt ICP for our system. Instead, during the stereo calibration for each system, the projector extrinsic matrix is kept as the identity matrix. This results in a global coordinate system that is shared between the two systems, since they share a projector. To merge the views into a single view, a Splat rendering (Rusinkiewicz and Levoy, 2000) is performed from the perspective of the projector, the coordinate origin. Once the coordinate maps and associated texture maps are splatted into a unified coordinate map, they are normalized based on the alpha value and then encoded with the Holovideo technique (Karpinsky et al., 2013b).

$$R = \frac{1}{2} + \frac{1}{2} * \sin(\frac{2\pi x}{P}), \tag{9.15}$$

$$G = \frac{1}{2} + \frac{1}{2} * \cos(\frac{2\pi x}{P}), \tag{9.16}$$

$$B = S * \left\lfloor \frac{x}{P} \right\rfloor - \frac{S-2}{2} * \cos\left[\frac{2\pi * mod(2,P)}{P_1}\right], \tag{9.17}$$

$$A = I', \tag{9.18}$$

where P is the fringe pitch used for encoding, S is the stair height, x is the X component of the fragment in the unit cube, P_1 is the local fringe pitch $P_1 = P/(K + .5)$, and K is the current fringe period.

At this point, all of the data has been processed and encoded, and is passed to the streaming module via another triple-buffer interface. Figure 9.6 shows the results of merging the left camera view and right camera view into a single frame.

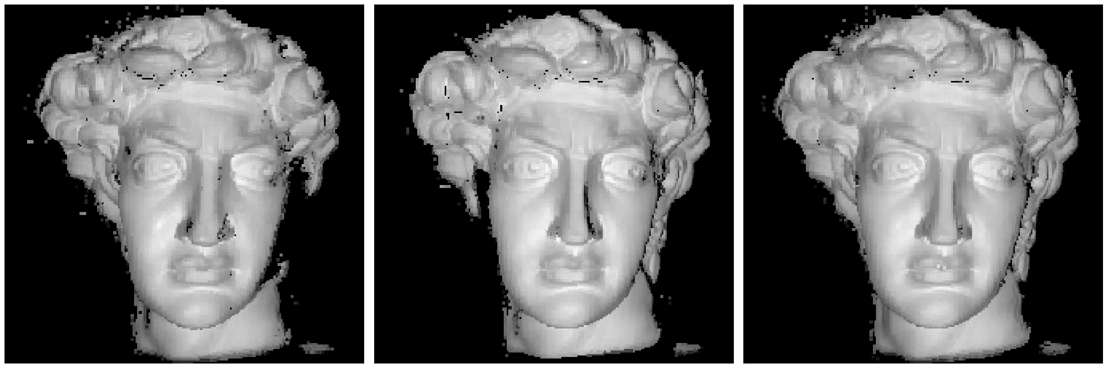

(a) (b) (c)

Figure 9.6 Example merging of the rigid calibration GPU Splat rendering merging.

Streaming the Holovideo encoded frames involves three steps: retrieving the frame from the GPU, compressing the image using a video codec, and then transmitting the result out onto the network. To retrieve the frame from the GPU, the streaming thread reads the Holovideo frame from the texture through another triple-buffer interface.

Once the frame is retrieved from the GPU it is video encoded; although Holovideo allows for motion JPEG or H.264 encoding we chose to utilize motion PNG. This is because the texture is also transmitted and neither of these video codecs support an alpha channel; this still reduces the three-channel single precision floating point coordinate map and texture, to a four-channel 32 bit image. Lastly, once encoded, the frames are sent out on a websocket to any connected clients. Details of the transmission and choice of protocol will be discussed in Section 9.4.2

9.4.2 Transmission

Transmission is an important design consideration to make, yet one that is often not covered in related literature. In order to illustrate the viability of such a system, one of the requirements is the ability to work over standard networks. Latency, bandwidth, and protocol overhead requirements are important to consider when selecting protocols to be used for the application. To ensure in-order data stream delivery, Portal-s uses TCP/IP as its transport protocol; this was chosen over UDP to guarantee in-order delivery of information in a fast manner, even though there is slightly more protocol overhead. In-order delivery is important because video encoders typically require in order streams; video encoders commonly use data from previous stream segments reducing temporal redundancy to reconstruct frames (Sullivan and Wiegand, 2005). Holovideo encoding has been shown to work with these formats (Karpinsky and Zhang, 2013), thus to be utilized in Portal-s the transmission module must support it. Secondly, websockets is the chosen communication protocol. It has been chosen as the communication protocol due to its low overhead and its ability to support full duplex communication through HTML5. This affords Portal-s flexibility to make use of technologies such as HTML5 and WebGL.

9.4.3 Display

The final module in Portal-s is display, in which 3D Holovideo encoded data coming in from the network needs to be decoded and displayed for the user. Similar to capture module, display has three major subsystems: receiving, processing, and display. Figure 9.7 shows the software architecture behind the display pipeline.

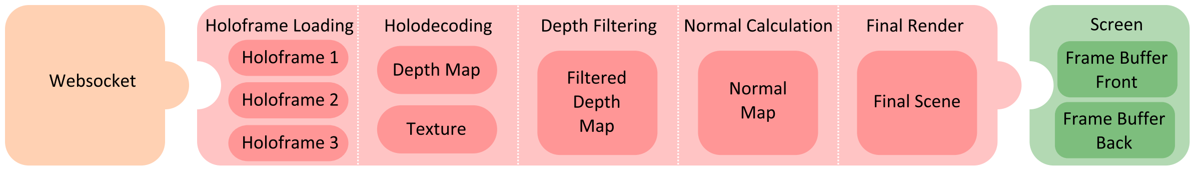


Figure 9.7 Display processing pipeline. Orange signifies the streaming thread, red signifies the display thread that is realized on the GPU, and green signifies the final system frame buffer display.

Receiving involves taking the encoded video, decoding into a Holovideo frame, and then passing it to the GPU. Once the frame is decoded, transfer to the GPU once again goes through a triple buffer interface to alleviate thread contention. Since the video stream is a live stream, playback rate does not need to be controlled, thus frames can be stream as soon as they are received.

Processing involves three stages, all of which are again performed on the GPU with GLSL image processing. The first stage, Holodecoding, involves decoding the depth map and texture from the Holovideo frame. The texture is simply the alpha channel of the image, and the depth is calculated via

$$\Phi = tan^{-1}\frac{I_r - .5}{I_g - .5} + floor(\frac{I_b}{S}) \times 2\pi, \tag{9.19}$$

$$Z = \frac{\Phi}{2\pi \times P}, \tag{9.20}$$

where Φ is the phase to be recovered, P is the fringe pitch or the number of pixels per fringe stripe, and S is the step height used in Eq 9.15-9.18. Next, filtering is applied to remove noise resulting from quantization and lossy encoding; this is done with the same 1×5 median filter shader that was used during capture to remove phase jumps, followed by a smaller 3×3 separable Gaussian filter. Next, point normals are calculated by calculating surface normals of adjacent points in a 3×3 neighborhood, and then normalizing them together into a normal map.

Display simply involves rendering the reconstructed depth map, as a plane of verticies with the Z value modified according to the corresponding depth value. The number of vertices is equal to the number of pixels in the decoded depth map, but it may be subsampled to improve performance if needed. Lastly, Phong shading along with the texture are applied to yield the final rendered result. This rendered scene, comes out of the visible light projector attached to the Portal-s node and is displayed on the Holographic display. The visible light projector is mounted below the screen at an angle of 38° perpendicular to the screen, so that (a) it does not occlude the IR structured light, and (b) so that it displays on the screen. The properties of the holographic display cause the light transmittance to decrease as the angle of projection is reduced. For the display used in our configuration 38° is the optimal angle. Due to the angle involved, either a short throw projector is needed, or a projector that allows for a large amount of key-stoning.

9.5 Results

To evaluate the ability of Portal-s to capture, transmission, and display modules, three different experiments were performed. In all of the experiments, each 2D camera was run at a resolution of 800×600 at 180 fps resulting in 3D reconstruction at 30fps. Holovideo encoding using an image size of 512×512 was utilized. The hardware consisted of Point Grey Research Flea3 cameras with a Computar 12 mm lens, a LogicPD Light Commander Projector equipped with a Sigma EX 24mm DG macro lens, and Vistique Hologlass from Vislogix. Finally, each Portal-s node was equipped with a Dell Precision

T7600 equipped with an Intel Xeon CPU at 2.4GHz and NVIDIA GeForce GTX 690. It should be noted that while the system can capture using the IR light, the intensity of the images are extremely low resulting in poor reconstruction. To remedy this, one of the nodes was switched to utilize white light to increase the intensity of the images resulting in better reconstruction.

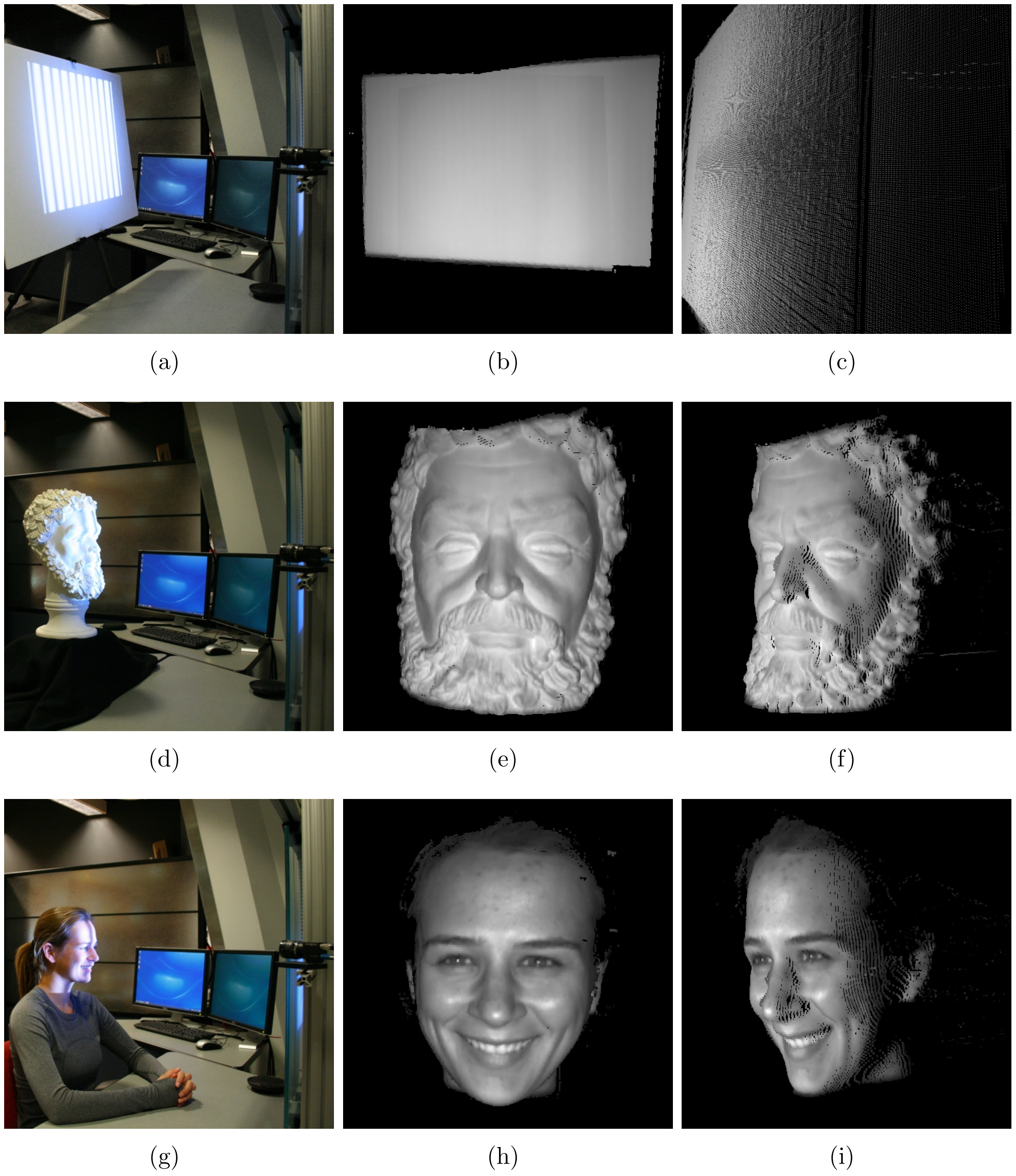
To evaluate the capture module, three different test objects were utilized: a flat reference plane, a static mannequin, and a live participant. Figure 9.8 shows the results of the objects placed in front of the system and corresponding reconstruction.

To evaluate the transmission module, we performed tests across a local high speed network, our campus network, and connecting to a local residential apartment through the local ISP. For a single frame, the worst case compression would result in 32 bits per pixel, equivalent to a bitmap image; this would require a throughput rate of 240Mbps, which can be achieved the local high-speed network but not on the others. With our live participant tests, average data throughput was 14Mbps, which allowed for data transfers across both the campus network and the local ISP at full frame rate.

Finally, evaluating the display module and its ability to achieve eye gaze was tested through informal discussions with users of the system. Although, formal users studies were not performed, users of the system have describe the system as being more natural than a Skype or FaceTime, and giving a feeling that the remote participant was in fact making eye contact. To quantify the results, further formal studies would need to be performed, possibly with eye tracking equipment to verify that eye gaze has in fact correctly been achieved.

9.6 Future Work

There are several avenues for future work on the system, from improving the performance of the system to utilizing it for activities in the field of computer supported



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 9.8 Evaluation of test objects within the Portal-s system. (Row 1) Flat reference plane (Row 2) static mannequin (Row 3) live participant.

collaborative work. Currently, the IR projector utilized in the system is dim, resulting in low intensity images with poor contrast; this results in increased noise in reconstruction. Improvements to this would result in higher quality captures, along with more resilience to noise artifacts in phase unwrapping. Color textures could be achieved by utilizing a color camera coupled with each IR camera Ou et al. (2013). This would result in more realistic reconstruction, possibly improving immersiveness but would require a correction mapping between camera sensors.

Finally, it would be worthwhile to conduct user studies, evaluating whether or not the system improves communication. Once such study would be to evaluate whether or not achieving correct eye gaze with the system improves communication and encourages a higher level of connection. Since the system involves a 3D rendering context, it is not merely limited to just presenting what the system is scanning. The context could be augmented with additional information, such as a part that is being working on by remote participants in an activity such as a design review; there are many such activities like this in the field of computer supported collaborative work that have yet to be fully realized.

9.7 Conclusion

In conclusion, the Portal-s 3D video conferencing system establishes correct eye gaze between users, allowing for mutual eye contact and a more natural interaction amongst users. This overcomes the primary challenge in current 2D video conferencing systems, the loss of eye contact. Through the use of a completely parallel 3D processing pipeline, dual camera structured light scanners are able to capture a wide field of view at 30 fps with a depth resolution of 0.5mm. A rigid calibration splat based merging allows the data to be merged in real-time while still on the GPU. Holovideo compression techniques allow the 3D data and texture to be streamed across a standard network and across a local internet service provider. Lastly, redisplayed is achieved on a holographic display whose display axis is aligned to the capture axis, allowing for corrected eye gaze. Future work include reducing spiking noise in the geometry, as well as creating a computer supported collaborative work environment by bringing in other information such as a shared desktop or shared 3D design.

Acknowledgments

This material is based upon work supported by the National Science Foundation (NSF) Graduate Research Fellowship under Grant No. DGE1247194 and the NSF under grant numbers CMMI-1150711 and CMMI-1300376. Any opinion, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the NSF. We would also like to acknowledge the Iowa State University (ISU) Institute for Physical Research and Technology (IPRT) for their funding support.

CHAPTER 10. General Conclusions and Future Work

In the previous chapters, current research in the development of real-time 3D scanning, 3D video compression, and their application to telepresence have been presented. This chapter summarizes the major contributions of this dissertation research and discusses potentials for extending this research.

10.1 Conclusions

This dissertation research has made substantial contributions to the fields of 3D scanning, 3D information processing, and 3D telepresence. The major contributions are:

1. **A framework to realize real-time 3D scanning entirely on a GPU.** A high resolution real-time 3D scanning framework that runs entirely on a GPU has been developed and demonstrated in this research. The system achieved 30 fps 3D scanning on an IBM Lenovo laptop with a Intel i5 3320M 2.6 GHz CPU and NVIDIA Quadro NVS5400M GPU. The real-time 3D geometry is recovered with a resolution of 800×600 pixels per frame. Utilizing a system geometry calibration, a phase jump were introduced into the equivalent frequency and unwrapped in parallel. To date, no other work has been able to utilize a GPU entirely due to serial computation in several steps of the pipeline.

2. **A novel method to substantially compress 3D geometry data.** Unlike models generated in 3D modeling or computer aided design software, models coming from a 3D scanner are inherently unstructured. This leaves traditional 3D file

compression ill-suited to compress such models. To address this challenge, a new compression technique that could compress 3D geometry into 2D images has been developed and demonstrated in this research. Once in the compressed 2D format, additional lossless or lossy image compression can be utilized to compress the geometry even further. The lossless compression ratio of 37:1 when compared to the OBJ file format was demonstrated.

3. **A technique to compress 3D video entirely on a GPU.** As real-time 3D scanning is realized on commodity hardware and massive amounts of 3D video are generated, a way of compressing the 3D video is needed. To address this, the Holovideo technique was developed and demonstrated in this research. The technique can compress 3D video in real-time on a GPU, and then further compress the video frames using either QuickTime video encoding or H.264 video encoding. Utilizing QuickTime video encoding, a compression ratio of 134:1 when compared to the OBJ file format was achieved with no noticeable compression artifacts. Utilizing H.264 video encoding, a lossless compression ratio of 352:1 when compared to the OBJ file format was achieved, and a lossy 6086:1 compression ratio was achieved.

4. **A method to interlace additional information with compressed 3D video.** Although the Holovideo technique can effectively compress 3D video coming from a real-time 3D scanner, many scans provide additional information such as 2D texture information. To accommodate this additional information, the Holovideo was extended with image dithering to further reduce the bit rate of the information. Through this reduction in bit rate, the additional information was interlaced into a single Holovideo frame, allowing 3D video and texture. The compression ratio of 8.2:1 when compared to standard Holovideo encoding was demonstrated, with a mean squared error of 0.34% while conveying both 3D video and associated 2D texture.

5. **Portal-s, a novel 3D video telepresence system to achieve correct eye gaze.** A set of algorithms and associated capture and display hardware entitled Portal-s were developed that established correct eye gaze between users. In traditional 2D video conferencing, a loss of correct eye gaze commonly occurs due to a disparity between the capture and display optical axes. Portal-s overcomes this problem through a dual camera structured light scanning system that captures through the display optical axis. We developed the holistic system, Portal-s, that achieves capture, transmission, and redisplay simultaneously at 30 fps 3D video across both local and wireless networks.

10.2 Future Work

Through this research many advancements have been made in the fields of 3D scanning, 3D information processing, and 3D telepresence. With these advancements, there are many venues for extending the work and applying it to new fields. Areas in which the technologies developed in this dissertation research could be further improved include:

1. **Real-time 3D Kalman filtering on GPU.** Currently, the real-time 3D geometry pipeline in this dissertation has made use of spatial filtering techniques such as Gaussian filters and median filters. Although these have led to high resolution reconstruction, one dimensionality of filtering has not be utilized, time. Currently, real-time Kalman filtering has been applied to medical imaging to assist in reconstructing MRI and CT scans. If such a filter could be creating using the OpenGL Shading Language and injecting into the 3D reconstruction pipeline, it could help alleviate noise due to high motion and low contrast scenes.

2. **Lossy video encoding on interlaced 3D geometry and texture Holovideo frame data.** While the Holovideo technique has been extended to interlace additional information such as 2D texture information, it must be kept in a lossless

state when doing this, so as not to incorrectly blend the data streams. Since most video codecs perform some level of lossy encoding on the data to achieve high levels of compression, it would be advantageous to extend the interlaced Holovideo technique to make use of them. Research into how to keep the data streams separate while sharing a Holovideo frame would need to be performed to do so. If this could be done, compression ratios of over 1000:1 when compared to the OBJ file format could be achieved.

3. **Color mapped 3D reconstruction in Portal-s.** To yield an immersive experience, Portal-s provides grayscale texture mapping onto the reconstructed 3D geometry. This approach suffers from two drawbacks, it is the infrared (IR) texture and it is only grayscale. To generate a more realistic reconstruction, it would be advantageous to have the correct color image mapped onto the 3D geometry. This could be done by augmenting the Portal-s node with additional color cameras next to the IR cameras, but would require more calibration and could possibly suffer from occlusion problems. The benefit of such an approach is that the color cameras would not pick up the IR structured light, thus they would capture the users natural ambient texture.

Applications that the technologies developed in this dissertation research could be immediately valuable to include:

1. **Computer supported collaborative work.** While the immediate benefit of achieving eye gaze in video conferencing can be seen with Portal-s, there are more activities that Portal-s could be extended to. One such activity is design review in the field of computer supported collaborative work. As Portal-s provides a full 3D environment for a user, additional information could be brought in such as a shared desktop. Users could collaborate over a document or presentation in the Portal-s environment while achieving correct eye gaze. Taking the concept of a

shared environment further, users could also collaborate over a 3D model such as one designed in a CAD program while achieving correct eye gaze. This would enable users to perform remote design reviews of 3D models and drawings.

2. **Large volume real-time 3D scanning.** Through the development of Portal-s, dual camera structured light scanners were merged together in real-time allowing for a wide field of view. With the use of the Portal-s 3D scanning engine, many structured light systems could be used simultaneously to scan much larger volumes. This opens the potential for high precision scanning over large areas, assisting in applications such as in situ inspection in manufacturing.

APPENDIX A. ADDITIONAL MATERIAL

BIBLIOGRAPHY

(2007). *NVIDIA CUDA Compute Unified Device Architecture - Programming Guide.*

Ahrenberg, L., Page, A. J., Hennelly, B. M., McDonald, J. B., and Naughton, T. J. (2009). Using commodity graphics hardware for real-time digital hologram view-reconstruction. *Journal of Display Technology*, 5(4):111–119.

Amorim, R., Haase, G., Liebmann, M., and Weber dos Santos, R. (2009). Comparing cuda and opengl implementations for a jacobi iteration. In *International Conference on High Performance Computing & Simulation*, pages 22–32. IEEE.

Asano, S., Maruyama, T., and Yamaguchi, Y. (2009). Performance comparison of fpga, gpu and cpu in image processing. In *International Conference on Field Programmable Logic and Applications*, pages 126–131. IEEE.

Aslantas, V. (2007). A depth estimation algorithm with a single image. *Optics Express*, 15(8):5024–5029.

Baldi, A. (2003). Phase unwrapping by region growing. *Applied Optics*, 42:2498–2505.

Bayer, B. (1973). An optimum method for two-level rendition of continuous-tone pictures. In *International Conference on Communication*, volume 1, pages 11 – 15. IEEE.

Beeler, T., Bickel, B., Beardsley, P., Sumner, B., and Gross, M. (2010). High-quality single-shot capture of facial geometry. *Transactions on Graphics*, 29(4):40.

Bernardini, F., Rushmeier, H., Martin, I., Mittleman, J., and Taubin, G. (2002). Building a digital model of michelangelo's florentine pieta. *Computer Graphics and Applications*, 22(1):59–67.

Besl, P. (1988). Active, optical range imaging sensors. *Machine Vision and Applications*, 1(2):127–152.

Besl, P. and McKay, N. (1992). A method for registration of 3-d shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256.

Briceño, H., Sander, P., McMillan, L., Gortler, S., and Hoppe, H. (2003). Geometry videos: a new representation for 3d animations. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 136–146. ACM.

Carpenter, J. and Wilkinson, T. D. (2010). Graphics processing unit–accelerated holography by simulated annealing. *Optical Engineering*, 49(9):095801–095801.

Carr, N., Hoberock, J., Crane, K., and Hart, J. (2006). Fast gpu ray tracing of dynamic meshes using geometry images. In *Proceedings of Graphics Interface*, pages 203–209. ACM.

Chen, M. (2002). Leveraging the asymmetric sensitivity of eye contact for videoconference. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 49–56. ACM.

Cheng, Y.-Y. and Wyant, J. C. (1985). Multiple-wavelength phase shifting interferometry. *Applied Optics*, 24:804–807.

Chyou, J.-J., Chen, S.-J., and Chen, Y.-K. (2004). Two-dimensional phase unwrapping with a multichannel least-mean-square algorithm. *Applied Optics*, 43:5655–5661.

Creath, K. (1987). Step height measurement using two-wavelength phase-shifting interferometry. *Applied Optics*, 26(14):2810–2816.

Cruz-Neira, C., Sandin, D., and DeFanti, T. (1993). Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 135–142. ACM.

Cuevas, F. J., Servin, M., Stavroudis, O. N., and Rodriguez-Vera, R. (2000). Multi-layer neural networks applied to phase and depth recovery from fringe patterns. *Optics Communications*, 181(4):239–259.

Curless, B. (1999). From range scans to 3d models. In *Proceedings of the 26st Annual Conference on Computer Graphics and Interactive Techniques*, volume 33 of *SIGGRAPH '99*, pages 38–41. ACM.

Davis, J., Ramamoorthi, R., and Rusinkiewicz, S. (2003). Spacetime stereo: A unifying framework for depth from triangulation. In *Conference on Computer Vision and Pattern Recognition*, volume 27, pages 1–7. IEEE.

Elseberg, J., Borrmann, D., and Nuchter, A. (2011). Efficient processing of large 3d point clouds. In *2011 XXIII International Symposium on Information, Communication and Automation Technologies*, pages 1–7. IEEE.

Espinosa-Romero, A. and Legarda-Saenz, R. (2011). Gpu based real-time quadrature transform method for 3-d surface measurement and visualization. *Optics Express*, 19(13):12125–12130.

Fang, J., Varbanescu, A. L., and Sips, H. (2011). A comprehensive performance comparison of cuda and opencl. In *International Conference on Parallel Processing*, pages 216–225. IEEE.

Favaro, P. and Soatto, S. (2005). A geometric approach to shape from defocus. *Transactions on Pattern Analysis and Machine Intelligence*, 27(3):406–417.

Floyd, R. W. (1976). An adaptive algorithm for spatial gray-scale. In *Proceedings of the Society for Information Display*, volume 17, pages 75–77. SID.

Flynn, T. J. (1997). Two-dimensional phase unwrapping with minimum weighted discontinuity. *Journal of the Optical Society of America A*, 14:2692–2701.

Forstmann, S., Ohya, J., Krohn-Grimberghe, A., and McDougall, R. (2007). Deformation styles for spline-based skeletal animation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 141–150. ACM.

Fung, J. and Mann, S. (2004). Computer vision signal processing on graphics processing units. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages V–93. IEEE.

Gao, W. and Kemao, Q. (2010). Real-time pipelined heterogeneous system for windowed fourier filtering and quality guided phase unwrapping algorithm using graphic processing unit. In *AIP Conference Proceedings*, volume 1236, page 129. AIP.

Gao, W. and Kemao, Q. (2012). Parallel computing in experimental mechanics and optical measurement: A review. *Optics and Lasers in Engineering*, 50(4):608–617.

Gao, W., Wang, L., and Hu, Z. (2008). Flexible method for structured light system calibration. *Optical Engineering*, 47(8):083602.

Geng, G. (2011). Structured-light 3d surface imaging: a tutorial. *Advances in Optics and Photonics*, 3(2):128–160.

Geng, Z. J. (1996). Rainbow 3-d camera: New concept of high-speed three vision system. *Optical Engineering*, 35:376–383.

Ghiglia, D. C. and Pritt, M. D. (1998). *Two-Dimensional Phase Unwrapping: Theory, Algorithms, and Software*. John Wiley and Sons, Inc.

Ghiglia, D. C. and Romero, L. A. (1996). Minimum l^p-norm two-dimensional phase unwrapping. *Journal of the Optical Society of America A*, 13:1–15.

Gorthi, S. and Rastogi, P. (2010). Fringe projection techniques: whither we are? *Optics and Lasers in Engineering*, 48(2):133–140.

Grayson, D. M. and Monk, A. F. (2003). Are you looking at me? eye contact and desktop video conferencing. *Transactions on Computer-Human Interaction*, 10(3):221–243.

Gross, M., Würmlin, S., Naef, M., Lamboray, E., Spagno, C., Kunz, A., Koller-Meier, E., Svoboda, T., Van Gool, L., Lang, S., Strehlke, K., Moere, A. V., and Staadt, O. (2003). blue-c: a spatially immersive display and 3d video portal for telepresence. *Transactions on Graphics*, 22(3):819–827.

Gu, X., Gortler, S., and Hoppe, H. (2002). Geometry images. *Transactions on Graphics*, 21(3):355–361.

Gu, X., Zhang, S., Huang, P., Zhang, L., Yau, S., and Martin, R. (2006). Holoimages. In *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling*, pages 129–138. ACM.

Guan, C., Hassebrook, L. G., and Lau, D. L. (2003). Composite structured light pattern for three-dimensional video. *Optics Express*, 11(5):406–417.

Gumhold, S., Kami, Z., Isenburg, M., and Seidel, H. (2005). Predictive point-cloud compression. In *ACM SIGGRAPH 2005 Sketches*, page 137. ACM.

Guo, H., He, H., and Chen, M. (2004). Gamma correction for digital fringe projection profilometry. *Applied Optics*, 43:2906–2914.

Guo, H. and Huang, P. (2008). 3-d shape measurement by use of a modified fourier transform method. In *Proceedings of SPIE*, volume 7066, page 70660E. SPIE.

Hartley, R. and Zisserman, A. (2000). *Multiple view geometry in computer vision*, volume 2. Cambridge University Press.

Hassebrook, D. L. (2010). Mat5 data format. Online: http://www.engr.uky.edu/ lgh/soft/softmat5format.htm.

Hocken, R. (1995). *Coordinate measuring machines and systems*, volume 42. CRC.

Hornbeck, L. J. (1997). Digital light rrocessing for high-brightness, high-resolution applications. In *SPIE Electronic Imaging*, volume 3013, pages 27–40. SPIE.

Hou, Z., Su, X., and Zhang, Q. (2012). Virtual structured-light coding for three-dimensional shape data compression. *Optics and Lasers in Engineering*, 50(6):844–849.

Hu, Q., Huang, P. S., Fu, Q., , and Chiang, F. P. (2003). Calibration of a 3-d shape measurement system. *Optical Engineering*, 42(2):487–493.

Huang, L., Chua, P. S. K., and Asundi, A. (2010). Least-squares calibration method for fringe projection profilometry considering camera lens distorsion. *Applied Optics*, 49:1539–1548.

Huang, P. S., Hu, Q., Jin, F., and Chiang, F. P. (1999). Color-encoded digital fringe projection technique for high-speed three-dimensional surface contouring. *Optical Engineering*, 38:1065–1071.

Huang, P. S., Zhang, C., and Chiang, F.-P. (2002). High-speed 3-d shape measurement based on digital fringe projection. *Optical Engineering*, 42(1):163–168.

Huang, P. S. and Zhang, S. (2006). Fast three-step phase shifting algorithm. *Applied Optics*, 45:5086–5091.

Huang, P. S., Zhang, S., and Chiang, F.-P. (2005). Trapezoidal phase-shifting method for three-dimensional shape measurement. *Optical Engineering*, 44:123601.

Hung, K. M. and Yamada, T. (1998). Phase unwrapping by regions using least-squares approach. *Optical Engineering*, 37:2965–2970.

Huntley, J. M. (1989). Noise-immune phase unwrapping algorithm. *Applied Optics*, 28:3268–3270.

Ishii, I., Yamamoto, K., Tsuji, T., et al. (2007). High-speed 3d image acquisition using coded structured light projection. In *International Conference on Intelligent Robots and Systems*, pages 925–930. IEEE.

Jones, A., Lang, M., Fyffe, G., Yu, X., Busch, J., McDowall, I., Bolas, M., and Debevec, P. (2009). Achieving eye contact in a one-to-many 3d video teleconferencing system. *Transactions on Graphics*, 28(3):64:1–64:8.

Jones, A., McDowall, I., Yamada, H., Bolas, M., and Debevec, P. (2007). Rendering for an interactive 360 light field display. *Transactions on Graphics*, 26(3):40.

Kakunai, S., Sakamoto, T., and Iwata, K. (1999). Profile measurement taken with liquid-crystal grating. *Applied Optics*, 38(13):2824–2828.

Kang, H., Yamaguchi, T., Yoshikawa, H., Kim, S.-C., and Kim, E.-S. (2008). Acceleration method of computing a compensated phase-added stereogram on a graphic processing unit. *Applied Optics*, 47(31):5784–5789.

Karpinsky, N. (2011). *3D geometry compression with Holoimage*. Iowa State University.

Karpinsky, N., Hoke, M., Chen, V., and Zhang, S. (2013a). High-resolution real-time 3d shape measurement on a portable device. In *Optical Engineering + Applications*, pages 88390K–88390K. SPIE.

Karpinsky, N., Wang, Y., and Zhang, S. (2013b). Three-bit representation of three-dimensional range data. *Applied Optics*, 52(11):2286–2293.

Karpinsky, N. and Zhang, S. (2010a). Composite phase-shifting algorithm for three-dimensional shape compression. *Optical Engineering*, 49(6):063604–063604.

Karpinsky, N. and Zhang, S. (2010b). High-resolution, real-time 3d imaging with fringe analysis. *Journal of Real-Time Image Processing*, pages 1–12.

Karpinsky, N. and Zhang, S. (2012a). Generalizing holovideo to h.264. In *SPIE Electronic Imaging*, volume 8290, page 829012. SPIE.

Karpinsky, N. and Zhang, S. (2012b). Holovideo: Real-time 3d range video encoding and decoding on gpu. *Optics and Lasers in Engineering*, 50(2):280–286.

Karpinsky, N. and Zhang, S. (2013). 3d range geometry video compression with the h. 264 codec. *Optics and Lasers in Engineering*.

Kauff, P., Atzpadin, N., Fehn, C., Muller, M., Schreer, O., Smolic, a., and Tanger, R. (2007). Depth map creation and image-based rendering for advanced 3dtv services providing interoperability and scalability. *Signal Processing: Image Communication*, 22(2):217–234.

Kite, T. D., Evans, B. L., and Bovik, A. C. (2000). Modeling and quality assessment of halftoning by error diffusion. In *International Conference on Image Processing*, volume 9, pages 909 – 922. IEEE.

Krishnamurthy, R., Chai, B., Tao, H., and Sethuraman, S. (2001). Compression and transmission of depth maps for image-based rendering. In *2001 International Conference on Image Processing*, volume 3, pages 828–831. IEEE.

Kunz, A. and Spagno, C. (2004). Construction of a three-sided immersive telecollaboration system. *Presence: Teleoperators & Virtual Environments*, 13(2):178–192.

Lampert, C. (1999). The world of large-area glazing and displays. In *Switchable Materials and Flat Panel Displays*, pages 2–11. SPIE.

Lange, R. and Seitz, P. (2001). Solid-state time-of-flight range camera. *Journal of Quantum Electronics*, 37(3):390–397.

Legarda-Sáenz, R., Bothe, T., and Jüptner, W. P. (2004). Accurate procedure for the calibration of a structured light system. *Optical Engineering*, 43(2):464–471.

Lei, S. and Zhang, S. (2009). Flexible 3-d shape measurement method using projector defocusing. *Optics Letters*, 34(20):3080–3082.

Lei, S. and Zhang, S. (2010). Digital sinusoidal fringe generation: Defocusing binary patterns vs focusing sinusoidal patterns. *Optics and Lasers in Engineering*, 48(5):561–569.

Leica (2012). Scanstation p20. http://www.leica-geosystems.com/en/Leica-ScanStation-P20_101869.htm.

Levoy, M. (1990). Efficient ray tracing of volume data. *Transactions on Graphics*, 9(3):245–261.

Li, Y., Zhao, C., Qian, Y., Wang, H., and Jin, H. (2010). High-speed and dense three-dimensional surface acquisition using defocused binary patterns for spatially isolated objects. *Optics Express*, 18:21628–21635.

Li, Z., Shi, Y., Wang, C., and Wang, Y. (2008). Accurate calibration method for a structured light system. *Optical Engineering*, 47(5):053604.

Liu, K., Wang, Y., Lau, D. L., Hao, Q., and Hassebrook, L. G. (2010). Dual-frequency pattern scheme for high-speed 3-d shape measurement. *Optics Express*, 18:5229–5244.

Lohry, W. and Zhang, S. (2013). Genetic method to optimize binary dithering technique for high-quality fringe generation. *Optics Letters*, 38(4):540–542.

Malacara, D. (2007). *Optical Shop Testing*. John Wiley & Sons.

Matusik, W., Buehler, C., and McMillan, L. (2001). Polyhedral visual hulls for real-time rendering. *Rendering Techniques*, pages 115–125.

Matusik, W., Buehler, C., Raskar, R., Gortler, S. J., and McMillan, L. (2000). Image-based visual hulls. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 369–374. ACM.

McGuire, M. (2008). *A fast, small-radius GPU median filter*. Cengage Learning.

Merráez, M. A., Boticario, J. G., Labor, M. J., and Burton, D. R. (2005). Agglomerative clustering-based approach for two dimensional phase unwrapping. *Applied Optics*, 44:1129–1140.

Merry, B., Marais, P., and Gain, J. (2006). Compression of dense and regular point clouds. *Computer Graphics Forum*, 25(4):709–716.

Microsoft (2010). Kinect. http://www.xbox.com/en-US/kinect.

Minsky, M. (1980). Telepresence. *Omni*, 2(9):45–52.

Möller, T., Haines, E., and Hoffman, N. (2002). *Real-time rendering*. AK Peters Limited, Natick, MA, USA, 2nd edition.

Morris, E. (2004). The fog of war: 13 questions and answers on the filmmaking of errol morris. In *FLM Magazine*.

Munshi, A. (2007). Opengl es common profile specification 2.0. *Khronos Group September*.

Munshi, A. et al. (2009). The opencl specification. *Khronos OpenCL Working Group*, 1:l1–15.

Ogale, A. and Aloimonos, Y. (2005). Shape and the stereo correspondence problem. *International Journal of Computer Vision*, 65(3):147–162.

Oggier, T., Lehmann, M., Kaufmann, R., Schweizer, M., Richter, M., Metzler, P., Lang, G., Lustenberger, F., and Blanc, N. (2004). An all-solid-state optical range camera for 3d real-time imaging with sub-centimeter depth resolution (swissranger). In *Optical Systems Design*, pages 534–545. SPIE.

Ou, P., Li, B., Wang, Y., and Zhang, S. (2013). Flexible real-time natural 2d color and 3d shape measurement. *Optics Express*, 21(14):16736–16741.

Pan, B., Kemao, Q., Huang, L., and Asundi, A. (2009). Phase error analysis and compensation for nonsinusoidal waveforms in phase-shifting digital fringe projection profilometry. *Optics Letters*, 34(4):2906–2914.

Pan, J., Huang, P. S., and Chiang, F.-P. (2006). Color phase-shifting technique for three-dimensional shape measurement. *Optical Engineering*, 45(12):013602.

Radiohead (2008). Radiohead: House of cards. Online: http://www.youtube.com/watch?v=8nTFjVm9sTQ.

Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., and Fuchs, H. (1998). The office of the future: a unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 179–188. ACM.

Remondino, F. (2011). Heritage recording and 3d modeling with photogrammetry and 3d scanning. *Remote Sensing*, 3(6):1104–1138.

Richardson, I. E. (2010). *The H.264 Advanced Video Compression Standard*. John Wiley & Sons.

Rohlf, J. and Helman, J. (1994). Iris performer: a high performance multiprocessing toolkit for real-time 3d graphics. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 381–394. ACM.

Rost, R. J., Kessenich, J. M., and Lichtenbelt, B. (2005). *OpenGL: Shading Language*. Addison-Wesley Professional, 2nd edition.

Rusinkiewicz, S., Hall-Holt, O., and Levoy, M. (2002). Real-time 3d model acquisition. *Transactions on Graphics*, 21(3):438–446.

Rusinkiewicz, S. and Levoy, M. (2000). Qsplat: A multiresolution point rendering system for large meshes. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 343–352. ACM.

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. In *Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. IEEE.

Salfity, M. F., Ruiz, P. D., Huntley, J. M., Graves, M. J., Cusack, R., and Beauregard, D. A. (2006). Branch cut surface placement for unwrapping of undersampled three-dimensional phase data: Application to magnetic resonance imaging arterial flow mapping. *Applied Optics*, 45:2711–2722.

Salvi, J., Fernandez, S., Pribanic, T., and Llado, X. (2010). A state of the art in structured light patterns for surface profilometry. *Pattern Recognition*, 43(8):2666–2680.

Salvi, J., Pages, J., and Batlle, J. (2004). Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827–849.

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1):7–42.

Schnabel, R. and Klein, R. (2006). Octree-based point-cloud compression. In *Symposium on Point-Based Graphics*, pages 111–120. ACM.

Schuchman, T. L. (1964). Dither signals and their effect on quantization noise. *Transactions on Communication Technology*, 12(4):162 – 165.

Shreiner, D. and Group, T. K. O. A. W. (2009). *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1.* Addison-Wesley Professional, 7th edition.

Stucki, P. (1981). Meccaa multiple-error correcting computation algorithm for bilevel hardcopy reproduction. Technical report, IBM Research Laboratory, Zurich, Switzerland.

Su, X. and Zhang, Q. (2010). Dynamic 3-d shape measurement method: A review. *Optics and Lasers in Engineering*, 48:191–204.

Subbarao, M. and Surya, G. (1994). Depth from defocus: a spatial domain approach. *International Journal of Computer Vision*, 13(3):271–294.

Sullivan, G. J. and Wiegand, T. (2005). Video compression-from concepts to the h. 264/avc standard. *Proceedings of the IEEE*, 93(1):18–31.

Takeda, M. and Mutoh, K. (1983). Fourier transform profilometry for the automatic measurement of 3-d object shape. *Applied Optics*, 22:3977–3982.

Towers, C. E., Towers, D. P., and Jones, J. D. (2003a). Optimum frequency selection in multifrequency interferometry. *Optics Letters*, 28(11):887–889.

Towers, C. E., Towers, D. P., and Jones, J. D. C. (2005). Absolute fringe order calculation using optimised multi-frequency selection in full-field profilometry. *Optics and Lasers in Engineering*, 43:788–800.

Towers, D. P., Jones, J. D. C., and Towers, C. E. (2003b). Optimum frequency selection in multi-frequency interferometry. *Optics Letters*, 28:1–3.

Tsai, R. (1987). A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *Journal of Robotics and Automation*, 3(4):323–344.

Upputuri, P. K., Mohan, N. K., and Kothiyal, M. P. (2009). Measurement of discontinuous surfaces using multiple-wavelength interferometry. *Optical Engineering*, 48:073603.

Vassilev, T. I. (2010). Comparison of several parallel api for cloth modelling on modern gpus. In *Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies*, pages 131–136. ACM.

Villa, Y., Araiza, M., Alaniz, D., Ivanov, R., and Ortiz, M. (2012). Transformation of phase to (x,y,z)-coordinates for the calibration of a fringe projection profilometer. *Optics and Lasers in Engineering*, 50(2):256–261.

Vislogix (2013). Vistique hologlass. http://vislogix.com/solutions/projection/transparent/vistique-hologlass/.

Wang, Y. and Zhang, S. (2011). Superfast multifrequency phase-shifting technique with optimal pulse width modulation. *Optics Express*, 19(6):5149–5155.

Weise, T., Leibe, B., and Van Gool, L. (2007). Fast 3d scanning with automatic motion compensation. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

Wen, Y., Li, S., Cheng, H., Su, X., and Zhang, Q. (2010). Universal calculation formula and calibration method in fourier transform profilometry. *Applied Optics*, 49(34):6563–6569.

Xiao, Y., Cao, Y., and Wu, Y. (2012). Improved algorithm for phase-to-height mapping in phase measuring profilometry. *Applied Optics*, 51(8):1149–1155.

Xu, Y., Ekstrand, L., Dai, J., and Zhang, S. (2011). Phase error compensation for three-dimensional shape measurement with projector defocusing. *Applied Optics*, 50(17):2572–2581.

Yang, R., Cheng, S., and Chen, Y. (2008). Flexible and accurate implementation of a binocular structured light system. *Optics and Lasers in Engineering*, 46(5):373–379.

Zhang, L., Curless, B., and Seitz, S. (2003). Spacetime stereo: Shape recovery for dynamic scenes. In *Conference on Computer Vision and Pattern Recognition*, pages 367–374. IEEE.

Zhang, L., Snavely, N., Curless, B., and Seitz, S. M. (2004). Spacetime faces: High-resolution capture for modeling and animation. *Transactions on Graphics*, 23(3):548 – 558.

Zhang, S.

Zhang, S. (2010). Recent progresses on real-time 3d shape measurement using digital fringe projection techniques. *Optics and Lasers in Engineering*, 48(2):149–158.

Zhang, S. and Huang, P. (2006a). High-resolution, real-time three-dimensional shape measurement. *Optical Engineering*, 45(12):123601–123601.

Zhang, S. and Huang, P. (2006b). Novel method for structured light system calibration. *Optical Engineering*, 45(8):083601–083601.

Zhang, S. and Huang, P. S. (2007). Phase error compensation for a three-dimensional shape measurement system based on the phase shifting method. *Optical Engineering*, 46(6):063601.

Zhang, S., Li, X., and Yau, S.-T. (2007). Multilevel quality-guided phase unwrapping algorithm for real-time three-dimensional shape reconstruction. *Applied Optics*, 46:50–57.

Zhang, S., Royer, D., and Yau, S.-T. (2006). Gpu-assisted high-resolution, real-time 3-d shape measurement. *Optics Express*, 14(20):9120–9129.

Zhang, S., van der Weide, D., and Olvier, J. (2010). Superfast phase-shifting method for 3-d shape measurement. *Optics Express*, 18(9):9684–9689.

Zhang, S. and Yau, S.-T. (2006). High-resolution, real-time 3d absolute coordinate measurement based on a phase-shifting method. *Optics Express*, 14:2644–2649.

Zhang, S. and Yau, S.-T. (2007a). Generic nonsinusoidal phase error correction for three-dimensional shape measurement using a digital video projector. *Applied Optics*, 46(1):36–43.

Zhang, S. and Yau, S.-T. (2007b). High-speed three-dimensional shape measurement using a modified two-plus-one phase-shifting algorithm. *Optical Engineering*, 46(11):113603.

Zhang, S. and Yau, S.-T. (2008). Three-dimensional data merging using holoimage. *Optical Engineering*, 47(3):033608.

Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the Seventh International Conference on Computer Vision*, volume 1, pages 666–673. IEEE.

Zhang, Z. (2000). A flexible new technique for camera calibration. *Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334.