Project Number: CF-RI13

Fully Reversed Engineering: streamlining 3D component digitization, modification, and reproduction

A Major Qualifying Project Report

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

in Mechanical Engineering

by

Ryan Muller

Chris Thomas

Date: April 25, 2013

Keywords:
1. structured light
2. rapid digitization

Approved:

Professor Cosme Furlong, Advisor

Abstract

The availability of rapid prototyping enhances a designer's creativity and speed, enabling quicker development of new products. However, because this process relies heavily on computer-aided design (CAD) models it can often be time costly and inefficient when a component is needed urgently in the field. This paper proposes a method to seamlessly integrate the digitization of existing objects with the rapid prototyping process. Our technique makes use of multiple structured-light techniques in conjunction with photogrammetry to build a more efficient means of product development. This combination of methods allows our developed application to rapidly scan an entire object using inexpensive hardware. Single views obtained by projecting binary and sinusoidal patterns are combined using photogrammetry feature tracking to create a computer model of the subject.

We present also the results of the application of these concepts, as applied to several familiar objects–these objects have been scanned, modified, and sent to a rapid prototyping machine to demonstrate the power of this tool. This technique is useful in a wide range of engineering applications, both in the field and in the lab. Future projects may improve the accuracy of the scans through better calibration and meshing, and test the accuracy of the digitized models more thoroughly.

Acknowledgments

The team would like to thank our advisor, Professor Cosme Furlong, for all his assistance throughout the project. His ideas, support, and genuine interest and expertise helped guide us throughout this long process. We thank him for his patience with our setbacks as well as his enthusiasm with our successes.

We would also like to thank Ivo Dobrev, Ellery Harrington, Babak Aghazadeh, and the rest of the members of the CHSLT Laboratory for their assistance throughout the year. Whether helping us find equipment in the lab, providing valuable insight on key concepts, or simply encouraging us to complete our goals, the lab members have been a tremendous support in our efforts.

Finally, we would like to thank PTI Industries for their funding and support. Their sponsorship gave us a clear goal for the project and allowed us a great deal of freedom in implementing our method.

Executive Summary

Introduction

Engineers are constantly seeking new tools to allow them to innovate more efficiently and effectively. The process of rapid digitization of an existing object, modification of that object in computer-aided design (CAD) software, and fabrication of the new object through rapid-prototyping is one such tool. Presently there are low cost options for both CAD packages and rapid prototyping. However, there does not presently exist a low cost solution for rapid-digitization of existing objects. Therefore, the goal of this project is to create a cost effective system for rapid digitization that is easily integrable with existing CAD packages and 3D printers.

There are many reasons why a designer or operator may wish to obtain a CAD model from an existing physical component or other object. These reasons include reproduction of an obsolete part, modification of a complex existing part, analysis of a worn or damaged part, and art conservation and restoration, among others.

Background

To develop our system, we analyzed all major optical imaging techniques that have been published, as well as the workings of some commercial scanning packages. Our research specifically placed emphasis on photogrammetry, a technique which recovers shape based on a series of photographs; spatially-multiplexed methods, a

group of techniques that can reconstruct a 3D image from a single camera frame; and temporally-multiplexed methods, a category of techniques that use a series of projected patterns to recover 3D shape from a static scene. Both spatial and temporal multiplexing techniques fall under the category of structured light; techniques which consist of projecting one or more encoded patterns onto an object and recovering shape data through triangulation.

Methods

Based on our goal of building a low-cost, accessible device, we decided that temporal multiplexing would be appropriate. This would allow us to use inexpensive hardware and yet still recover high-quality results.

The novelty of our approach lies in the combination of two structured light techniques. The first uses binary-coded patterns to quickly correlate pixels in the camera image to pixels in the projected pattern. This allows for generation of an approximation of the mesh within a few seconds. This approximation, though, is very rough, and is unsuitable for most engineering applications.

The second technique, called phase-shifted fringe projection, uses patterns that vary sinusoidally in intensity. By projecting different variants of the same pattern, a computer can recover the encoding with subpixel accuracy. However, there are three major drawbacks to this approach. First, this approach typically requires expensive hardware; second, the algorithm is computationally expensive, leading to a slow program; and third, the recovered information is periodic and lacks information about each pixel's absolute value, defining it only relative to other pixels in the same region of the photo.

To solve these problems, we implemented phase-shifting with a higher number of samples than the standard four images, allowing us to correct for signal noise by

averaging. Once our algorithm has collected the phase information, it uses the quick binary scan to "unwrap" the periodic phase map into a smooth representation of the distortions of the projected pattern on the object.

Implementation of our scanning process required an active projector and two or more digital cameras, and we used simulated results to help us select appropriate hardware. The projector was selected to be low cost, compact, and have a high resolution at the working distance. This last point is important because the accuracy of the scanning system is limited by the resolution of the projector; selecting a high-resolution projector with a short throw and a large throw ratio optimized our ability to measure accurately. Taking all of these parameters into account, the projector that was selected is the Optoma PK201 Pico Pocket Projector.

The cameras for the scanning system were selected to be low-cost and high-resolution. The spatial resolution of the cameras needs to be equal to or greater than the spatial resolution of the projector at the working distance. For this reason, the Logitech C905 webcam was selected

A custom mounting assembly was created for the setup, consisting of an L-beam, and custom designed camera and projector mounts. Additionally the camera casings were redesigned to fit the system. The system was designed to have flexible geometry, but also so the components could be locked in place so as to not change the calibration parameters. The whole assembly was placed inside a commercial light box in order to reduce stray light.

Scanning and duplicating a physical object requires digitization, format conversion, revision, and manufacture. Our research and implementation focus on digitization, which requires techniques for hardware setup, data capture, and orientation tracking.

The setup process for scanning is the preparation of the equipment. The cameras and projector must be placed in a triangulation geometry. Optimal positioning will

improve results and is determined by size of the object and the desired accuracy and precision. Calibration is then used to determine both the intrinsic and extrinsic parameters of the cameras and projector. First the cameras are calibrated, followed by the projector calibration.

Our data capture technique involves first capturing images of patterns projected onto the subject. Next, our algorithms process the image data into meaningful encodings. Finally, using these encodings, the hardware geometry, and our newly-developed "hybrid" algorithm, we reconstruct the object as a computer model. This computer model is represented by a point cloud, where each point represents a pixel from the camera image. To export to the popular STL format, we use a meshing technique where points are connected based on their adjacency in the captured image. After export, the STL model may be used in many popular software packages for the purposes previously discussed.

Results

Before implementation, to prove the viability of our scanning method, we simulated the binary technique using POV-Ray raytracing software. This allowed us to specify hardware characteristics and generate an image as if we had used a particular camera and projector. A theoretically perfect 15 centimeter radius sphere was modeled and processed into a point cloud. Cross sections of the point cloud were taken and radii were fitted to the cross sections. The average radius 3.2 micrometers less than expected, with a standard deviation of 20.75 micrometers. The data also revealed that the system was resistant to directional influence.

Our most significant result is the implementation of the software itself. Our software package, *Pico Scan*, allows the user to generate a point cloud of one side of an

object with a single click. The point cloud is displayed in 3D within the software itself. *Pico Scan* exports into the popular STL mesh format.

The final results achieved by this project were actual scans. These scans consisted of only a single 3D view of the object, and had noise caused by stray light in the regions where the patterns were not projected. For this reason the scans needed post-processing and manual stitching which prevented complete automation. However, the hybrid scans showed significant improvement in precision over the binary scans, validating the method.

Conclusions

This project set out to create a low cost system to digitize existing objects in full 3D for the purpose of enhancing the engineering design cycle. At the end the results we achieved proved that this is indeed possible with current technology and that with only a little more work these goals could have been fully satisfied. The hardware that was selected met the needs of the system and were sufficient to achieve engineering quality scans. Additionally the hybrid method that was proposed produced results that met the goals of the project. The only holdup was with the software development. For this reason it is recommended that future iterations of this project contain at least one computer science major to facilitate the implementation of software improvements.

Despite the fact that the software is not yet capable to realizing the full goal of the project, the theory is sound. The project was still a success because it proved that relatively low-cost hardware was capable of producing scans that could be used for engineering design or analysis. Additionally the current system is capable of producing the desired results, just not in an automated manner.

Future Work

- Finish projector calibration
- Implement stereo cameras
- Implement photogrammetry feature tracking
- Add noise filters
- Improve meshing algorithm
- Optimize software
- In-depth analysis of system limitations

Table of Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

α	horizontal scaling factor
β	vertical scaling factor
$\mathbf{D}$	distance vector between camera and projector
$\mathbf{M}$	measured point
u_c	horizontal location of pixel in camera image
u_p	horizontal location of pixel in projector image
v_c	vertical location of pixel in camera image
v_p	vertical location of pixel in projector image
CAD	computer-aided design
CCD	charge-coupled device
CMM	coordinate measuring machine
DLP	digital light processor
FEA	finite element analysis
FFT	fast-Fourier transform
PCL	Point Cloud Library

Chapter 1

Introduction

Engineers are constantly seeking new tools to allow them to innovate more efficiently and effectively. The process of rapid digitization of an existing object, modification of that object in computer-aided design (CAD) software, and fabrication of the new object through rapid-prototyping is one such tool. Presently there are low-cost options for both CAD packages and rapid prototyping. For example the open source software FreeCAD can be acquired free of charge. Additionally the RepRap project is a movement designed to make self replicating 3D printers so that as many people as possible can own one [RepRap, 2013]. While there are still some parts that can not be printed such as the extrusion head, the fact that people are able to print the majority of parts for their 3D printers makes them readily accessible. However one part of this process is currently lacking. There does not presently exist a low cost solution for rapid-digitization of existing objects. Therefore the goal of this project is to create a cost effective system for rapid digitization that is easily integrable with existing CAD packages and 3D printers.

Many engineering applications require digitized knowledge of the shape of a component or surface. The traditional method for obtaining this information, by use of a coordinate measuring machine (CMM), is costly, time consuming, and only provides

a sparse data set. Accordingly, there has been a proliferation of techniques, devices, and software allowing users to generate a computer model by quickly "scanning" a component by optical techniques.

These techniques still present challenges to small companies and hobbyists, since they are generally proprietary and often costly. This paper details the construction and use of an inexpensive, full field-of-view measuring system using easily obtained, commercially available components.

1.1 Importance of full field-of-view digitization

There are many reasons why a designer or operator may wish to obtain a CAD model from an existing physical component. To demonstrate the need for this technology, we will present several cases which represent a cross-section of the application of this type of system.

1.1.1 Replication or modification of existing parts

There are many situations where CAD files are not available for an existing component. In these situations sometimes it is infeasible to create a CAD through CAD software due to time constraints or complex geometry. In the situation of complex geometry in particular it is difficult to get precise measurements of 3D curves even with the use of a CMM. For example, Jay Leno uses an expensive 3D scanner to generate CAD models of antique car parts that are no longer commercially available. He is then able to create new replacement parts based on these models [Leno, 2011]. There may also be situations where the component to be replicated needs to be modified in some way. Perhaps the whole part needs to be scaled to another size, or an additional feature is required. Digitization of the part allows for the CAD file to be modified, meaning that such changes are possible.

1.1.2 In-the-field analysis of structural integrity

Sometimes there are situations where a part is damaged and a replacement is not readily available. In these scenarios it is often unclear as to whether or not it is safe to continue using the damaged part. Using such a part without proper knowledge of its structural integrity could lead to catastrophic failure, injury, and even death. By digitizing a damaged part with extensive detail on the damaged portion it is possible to create a CAD file with the damaged geometry. Finite element analysis (FEA) can then be performed on the model to see whether it is still safe to use or needs to be replaced immediately.

Additionally digitization can be used on a part that has already failed catastrophically. In this case the only important information is the digitized surface of the damaged region. By conducting a detailed analysis on the data obtained information can be gained on the failure mode. This will assist in avoiding repeated failure of replacement parts through either part redesign or maintenance and environmental changes.

1.1.3 Art conservation

Artwork is often created without any special attention paid to making it resistant to the flow of time. Even when preserved in museums with controlled conditions paintings and sculptures still undergo slow deterioration. With digitization technology it is possible to preserve sculptures in digital archives for viewing online by future patrons. Additionally having a record of the current state of artwork would allow for restoration efforts to more accurately recreate the original.

Figure 1.1: WeatherTech liners [WeatherTech, 2013].

1.1.4 Custom accessories

Sometimes rather than modifying an existing part it is easier to create an accessory to enhance its function. In these scenarios it is often necessary to know the precise geometry of the existing part to which the accessory will be attached. For example the company WeatherTech uses digitization technology to create custom floor liners for cars. These floor liners are created to be form fitting in order to not allow any water to harm to car interior. Thus the company first digitizes the region of the interior that the liners will be installed in, and uses this information to generate the exact geometry of the mats themselves [WeatherTech, 2013]. A similar process could be applied to any situation where a form fitting accessory is required. Figure 1.1 shows examples of the WeatherTech liners.

1.1.5 Quality assurance

Digitization technology can be implemented in the manufacturing process. Rapid digitization can be used on an assembly line to scan parts. Basic feature recognition can be used on the digitized parts to check for defects. Even if the process is not fast enough to scan a part as its moving on a conveyor, it can still be used on a sampling of parts for batch checking.

Chapter 2

Background

2.1 Three-dimensional optical shape measurement techniques

There are many different optical techniques for three-dimensional shape measurement. The techniques vary in range and applicability, and consequently are used in different professional fields. This Section gives a brief overview of the most common techniques available for three-dimensional shape measurement.

2.1.1 Photogrammetry

Photogrammetry is a passive technique that constructs a three-dimensional image from several two-dimensional photos. In order to achieve this, photogrammetry correlates features, colors, or patterns between photos. Photogrammetry techniques may take advantage of reflectivity, shading, and focus to recover shape information [Snavely, 2008]. Being a passive technique, photogrammetry may be performed in ambient light without any illumination equipment. However, this method has much lower accuracy than most other methods, and therefore is generally not used in engineering or medical fields.

2.1.2 Time of Flight

This method directly measures the time of flight of a light source, typically a laser. The amount of time between the light being emitted, reflected off the object, and then received by the sensor is used to calculate the distance from the camera to each point on the object [Chen et al., 2000]. Time of flight techniques have a much longer range than most other shape measurement techniques, but they are also lower resolution. This makes it useful for surveying, and other long range purposes.

2.1.3 Triangulation

Triangulation techniques use the geometry of a projector and camera to reconstruct a three-dimensional object as a computer model. In its simplest form, a system projects a pattern and then views that pattern from a different angle as it deforms on the object. Using the focal lengths of the camera and projector as well as their relative positions and orientations, such a system can recover the original shape through processing of the distorted pattern Hartley and Sturm [1997].

2.1.3.1 Laser Scanning

In its simplest form, a laser scanner operates by projecting a dot onto the subject and calculating the three-dimensional position of that dot from its captured camera image. Typically, this process is accelerated through the use of a laser line instead of a dot, and occasionally multiple lines are used to speed the process further [Chen et al., 2000]. Image processing techniques, such as derivative filters, may be used to locate the center of brightness of the laser line and allow resolutions greater than the thickness of the line would allow. Figure 2.1 shows the basics of a line scanner.

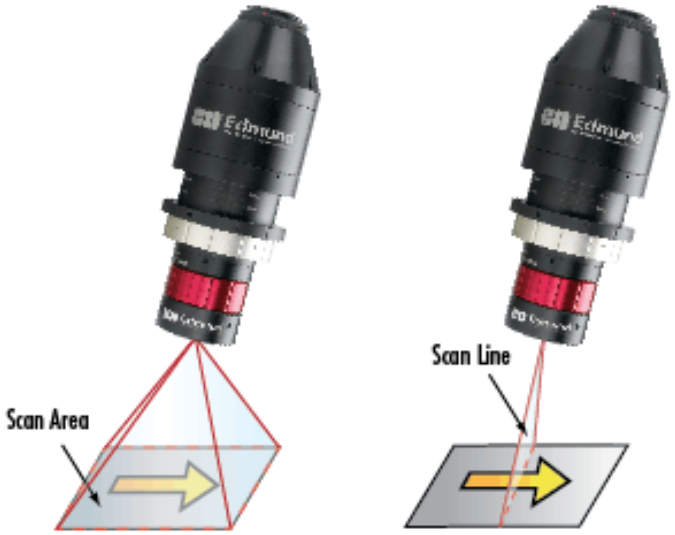


Figure 2.1: Laser scanner (a) scanning volume; (b) scan line [Edmund Optics, 2012].

2.1.3.2 Structured light

Structured light techniques use a coded pattern of projected light in conjunction with a camera to quickly perform full field-of-view triangulation. Depending on the patterns used, a technique may be classified as either continuous or discrete. Continuous coding techniques include sinusoidal fringe projection and color coded fringe projection, while discrete techniques include spatial multiplexing, and time multiplexing [Salvi et al., 2004].

2.1.3.2.1 Continuous coding Continuous coding is a term for any structured light technique that projects a continuous pattern in order to code the shape data into an image. Most continuous coding techniques utilize a sinusoidal pattern, but there are some that use other forms of continuous information. Continuously-coded images contain information about the projected pattern at every pixel of the camera image [Salvi et al., 2010].

Sinusoidal fringe projection Sinusoidal fringe projection is a category of techniques that use a projection pattern that varies sinusoidally in intensity. A static fringe pattern may be used when rapid image acquisition is necessary, as in analysis of a moving assembly. These frames may be analyzed using Fourier transformation theory to very quickly reconstruct an approximate model. This method suffers from errors when the subject's profile contains step discontinuities, however, and therefore is best used to characterize a single surface.

Fourier transform profilometry is a method for calculating the wrapped phase map of the object in a single frame. The method takes the Fourier transform of the intensity and isolates the shape containing phase information. When the Fourier transform is performed, 3 distinct peaks result in the Fourier domain. The central peak contains the brightness information and can be masked out. The two remaining peaks are symmetric about the origin and contain the shape information. One of these peaks is masked out, and the remaining one is shifted by the carrier frequency to the origin [Takeda and Mutoh, 1983]. The inverse fast-Fourier transform (FFT) is calculated and the phase data is separated from the contrast by taking the arc tangent of the imaginary components over the real components [Zervas, 2011].

When speed of scanning is not paramount, another commonly-used technique is phase shifting, which involves projecting multiple fringe patterns with a specific phase difference between them. When these images are captured, they may be transformed using a least-squares equation at every point in the image to capture the original phase data, corresponding to the angle between the projector's optical axis and the projected pixel. This phase map, containing what is known as "wrapped" data, must be "unwrapped" before three-dimensional data can be extracted. This can be accomplished through a "flood-fill" method, through Fourier transformation, or through use of multiple fringe densities [Zervas, 2011].

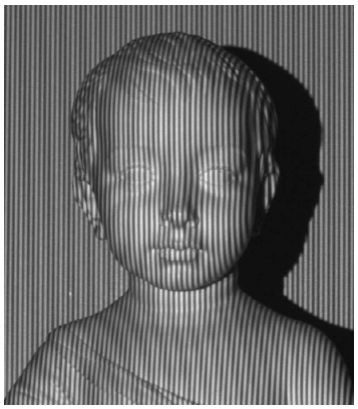


Figure 2.2: Sinusoidal fringe projection [Zervas, 2011].

It is important when using either of these techniques that the fringes be as close to sinusoidal as possible. One technique to ensure accurate sinusoids is to use a digital light processor (DLP) projector with a high frame rate. Such a system can rapidly project multiple binary patterns during the exposure of one frame of the camera and very accurately recreate a sinusoidal pattern. To reduce costs, other systems approximate sinusoidal fringes by using a single, defocused binary pattern. This simplifies implementation, as the hardware need only be capable of projecting binary patterns, but introduces errors as the approximation deviates from a true sinusoid. Figure 2.2 shows sinosoidal fringes projected on a statue.

Color-coded fringe projection Instead of a single fringe pattern being phase shifted between several pictures, three phase shifted patterns are projected simultaneously for single frame acquisition. These three patterns are different colors, typically red, green, and blue as these are the primary colors of any commercial projection system, as seen in Fig. 2.3. The three patterns are separated and used to generate a wrapped phase map. The main limitation of this method is that it is sensitive to the transmittance, reflectivity, and absorption of the object being measured. To compensate for this the exact wavelengths used can be chosen based on the color

and material of the object, or some form of coating can be applied to the object to improve the conditions [Huang et al., 1999].

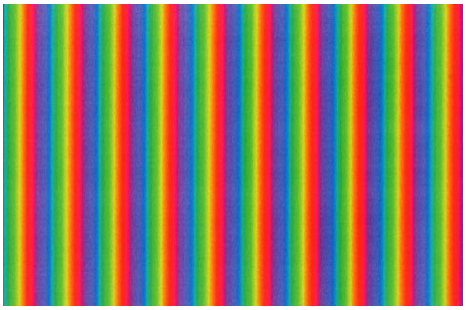


Figure 2.3: Color coded fringes [Huang et al., 1999].

Continuous spatial grading A continuous grayscale or color scale is projected onto the object. Every x-coordinate in the undistorted projection has a unique intensity value, allowing for triangulation similar to a line scanner. This method is extremely sensitive to shadowing and the color of the target object [Salvi et al., 2010].

2.1.3.2.2 Discrete coding Discrete coding consists of projecting non-continuous patterns onto an object. These patterns are designed such that every part of the image is uniquely identified by the pattern. This identification is referred to as the codeword for that location. The locations identified by codewords can either be lines or pixels, depending on whether the pattern is 1D or 2D. Since the location of each codeword is known in the projected image the displacement of the codeword when the pattern is projected on the object can be measured. This displacement can be used to triangulate the distance to the object for each location, thus giving the 3-dimensional shape. The two methods of discrete coding are spatial multiplexing and time multiplexing [Salvi et al., 2010].

2.1.3.2.3 Spatial multiplexing Spatial multiplexing methods project only a single pattern. The "codeword" for any given section of the pattern is based on the

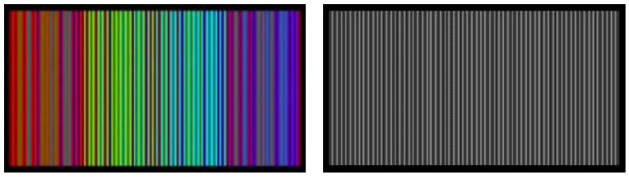


Figure 2.4: De Bruijn patterns [Salvi et al., 2010].

surrounding region. For a one-dimensional pattern this means the sequence of lines to either side of any given line are unique and thus identify that line. In a 2D case it the surroundings in all directions within the plane are taken into account [Salvi et al., 2004].

De Bruijn coding A pattern is constructed using a pseudorandom sequence known as a De Bruijn sequence. The properties of a De Bruijn sequence ensure that any projected line can be identified by the bordering lines, allowing for triangulation. This pattern can be binary (similar to bar code), grayscale, or color [Salvi et al., 2004]. Figure 2.4 is an example of a De Bruijn pattern.

M-arrays M-arrays are a 2D equivalent to the 1D De Bruijn patterns. An array of pseudorandom dots is projected onto the target object. Any dot can be identified by the adjacent dots, allowing for triangulation. This method can utilize binary, color or grayscale. The codeword of the dot can be identified not only by the type of dots around it, but by the relative density of the dots as well. [Salvi et al., 2010] An example M-array can be viewed in Fig. 2.5.

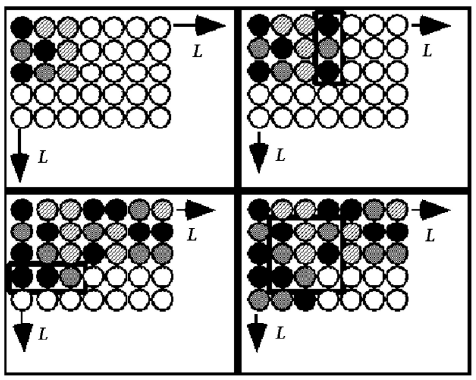


Figure 2.5: M-array [Salvi et al., 2010].

Non-formal coding Non-formal coding is a term used to categorize any number of spatial multiplexing methods that use unique patterns for specific purposes. These patterns do not necessarily directly identify the line or pixel as the previous methods do. Instead non-formal coding usually serves a more specific purpose such as calibration patterns [Salvi et al., 2004].

2.1.3.2.4 Time multiplexing Time multiplexing captures successive images with different patterns in order to generate the necessary codeword for each location. The patterns are generated such that each location has a unique sequence of values throughout the series of images. Since multiple frames are needed to generate a 3D image, this method is not viable for high speed applications that require single frame acquisition [Salvi et al., 2010].

Binary codes Binary codes function by projecting a series of binary patterns. These patterns are typically vertical lines of varying thicknesses or densities, similar to a bar code. A single pattern alone does nothing, but by taking into account the binary value of each line or pixel over the entire series of projected patterns, uniqueness is established [Salvi et al., 2004] . This is the primary technique used in our approach and will be discussed further in Section 3.3.

N-array codes N-array codes utilize the same basic concept as binary codes, except they are not restricted to binary patterns. They can utilize color or grayscale patterns to greatly reduce the number of frames necessary to uniquely identify each location in the image [Salvi et al., 2004].

Hybrid coding Hybrid coding consists of a combination of spatial and time multiplexing. Several spatial multiplexing patterns are displayed in series so as to create a time multiplex. This method achieves the high accuracy of time multiplexing, while greatly reducing the number of patterns necessary [Salvi et al., 2010].

2.1.4 Interferometry

Interferometry utilizes a beam splitter to separate a single beam into two beams. One of the beams, the sample beam, is reflected off the target object and into a sensor. This beam then meets with the other beam, the reference beam, in an interferometer. The interference between these two beams gives the phase difference of the lasers. The phases from all the points on the object are combined into a wrapped phase map image of the object [Chen et al., 2000]. The phase maps are then unwrapped, giving the shape of the object.

There are many advanced imaging techniques that use interferometry as a basis to generate absolute 3D measurements. One such technique is called laser speckle pattern sectioning. This method projects a speckle pattern on the target object which is measured using a charge-coupled device (CCD) array. The pattern is scanned through a range of wavelengths. Each wavelength corresponds to a 2D slice of the 3D object. By adding these slices together into a 3D data array, and then performing a 3D Fourier transform, the 3D shape can be found [Chen et al., 2000].

Interferometry has higher resolution and accuracy than many of the other techniques, and can be performed on a large range of object sizes depending on the setup.

For this reason it can be used in a large variety fields, making it a versatile technique [Z Corporation, 2012].

2.2 Existing commercial products

There are many different commercial scanners currently on the market. Most of them cater to different professional fields and have specifications that fit their application. A selected products are discussed in this Section.

2.2.1 Kinect

The Kinect is a 3D imaging device made by Microsoft for use with their Xbox 360 gaming console, shown in Figure 2.6 (a). The Kinect works based on a fixed pseudorandom array of dots projected on an infrared wavelength. The dots are formed by an array of small micro lenses, each with a slightly different focal length. This pattern can be seen in Fig. 2.6 (b). The included infrared camera picks up the projection of these dots on their environment. Groups of dots are then compared against an image taken on a reference plane. Due to the pseudorandom nature of the dot array, each group is unique enough to allow identification of a particular dot based on the relative positions of neighboring dots. Furthermore, due to the different focal lengths of the micro lenses, the pattern itself will vary based on the distance between the camera and the object [Freedman et al., 2009].

Microsoft has kept its specific algorithms for calculating the depth proprietary. However, the open-source community has had some success in reverse-engineering the Kinect. In its operating range between 0.8 and 3.5 meters, the Kinect can resolve depth with about 10 mm accuracy along the optical axis, and position to about 3 mm perpendicular to the optical axis [OpenKinect community, 2013].

(a) (b)

Figure 2.6: Microsoft Kinect. (a) the imaging device [Microsoft, 2012]; (b) projected pattern as seen through an infrared camera [Bernin, 2010].

2.2.2 Next Engine

Next Engine is a device that projects multiple laser lines onto the target object. To construct a 3D image of an object it performs line scanning in both the vertical and horizontal directions. It takes about two minutes to create a single 3D point cloud of the object. The software that is bundled with this product has the capacity to stitch together multiple views to create full 3D images. The 3D images are in full color and can be output to several common CAD formats. The Next Engine Scanner is marketed for use in design, manufacturing, CGI, art, and medical applications. This system boasts accuracy to 0.005 inches in macro mode and to 0.015 inches in wide mode [NextEngine Inc., 2012]. Figure 2.7 displays this system.

2.2.3 David 3D Laser Scanner

David 3D Laser Scanners come in two types. The first is a line scanning method that uses a line laser pointer and a digital camera. The laser pointer is scanned across the object by hand while the camera captures the image data. The other scanner is a sinusoidal fringe projection system. This scanner comes with calibration patterns and a software program capable of creating and stitching 3D point clouds. The scanner has a object size range from 10 to 600 millimeters with a accuracy up to 2% of the

Figure 2.7: Next Engine 3D scanner [NextEngine Inc., 2012].

object size. It takes 2 to 4 seconds per scan and generates grayscale images [David LaserScanner, 2012].

2.2.4 Handy Scan 3D

The Handy Scan 3D scanner is a portable line scanner. It boasts an accuracy of up to 40 microns. The Handy Scan projects a cross hair onto the target object

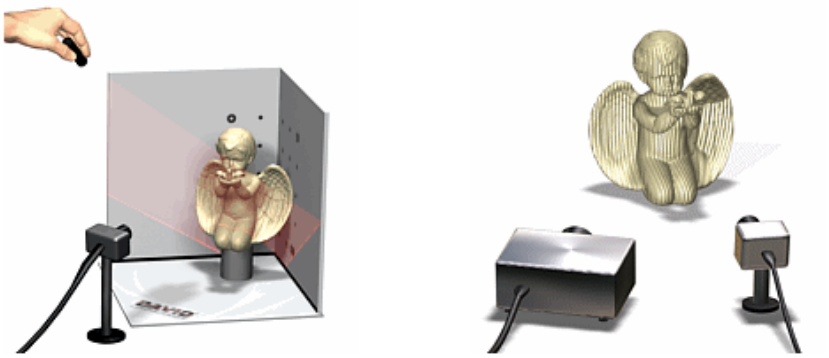


Figure 2.8: David 3D Laser scanners [David LaserScanner, 2012].

Figure 2.9: Handy Scan 3D scanner [Creaform, 2012].

and scans in both x and y simultaneously. The device has a camera built into it so the triangulation geometry remains constant as the laser is scanned along the object. The technology requires several sensors to be placed on the object. These sensors are randomly placed on the object and are triangulated by two cameras on the scanner. This allows the scanner to know its location relative to the object, making the freehand scanning possible. This product is marketed for reverse engineering, design, and part inspection [Creaform, 2012]. This product is shown in Fig. 2.9.

Chapter 3

Methods

3.1 Overview of Scanning Process

The process of scanning a real world object and using it to make a new object can be broken into 4 distinct steps. These steps are digitization, conversion to mesh, manipulating in CAD software, and manufacturing. Figure 3.1 contains a flowchart detailing this process. Each of the steps will be explained in further detail below.

3.1.1 Digitization

Digitization refers to the process of converting the dimensions of the 3-dimensional object into data to be used by computer software. This process can be broken into three distinct sections; setup, data capture, and repositioning, as seen in Fig. 3.2

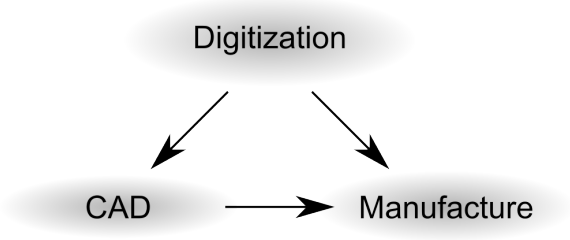


Figure 3.1: Scanning process overview.

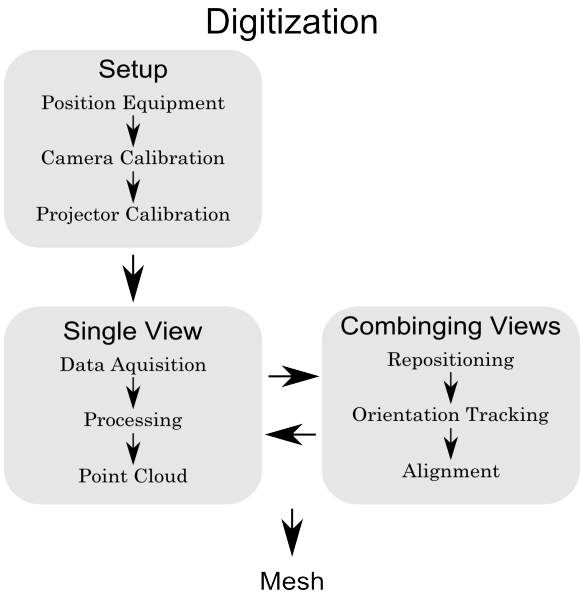


Figure 3.2: Digitization process overview.

3.1.1.1 Setup

The setup process for scanning is the preparation of the equipment. For this scanning process the necessary equipment are an active projector and two or more digital cameras. The quality of these pieces of equipment are determined by the quality of the results required. The cameras and projector must be placed in a triangulation geometry. Optimal positioning will improve results and is determined by size of the object and the desired accuracy and precision.

Once the equipment is positioned it is necessary to calibrate it. Calibration is used to determine both the intrinsic and extrinsic parameters of the cameras and projector. This means that the calibration serves to both remove the distortions from the lenses of the devices, but also to establish the relative locations and orientations of the different pieces of equipment relative to each other. First the cameras are calibrated simultaneously, and then the projector is calibrated. Detailed descriptions of the calibration procedure can be found in Section 3.2.

3.1.1.2 Data Capture

Data capture can be broken into three distinct steps; data acquisition, data processing, and point cloud generation. The data acquisition is achieved by taking images with the two cameras. In the case of this method 20 frames of data are needed for each data capture view. Once the data is squired the images are correct for distortions using the calibration data and are then processed using the triangulation algorithm. Once each pixel is processed a point cloud is constructed using the x, y, and z information obtained from the calibration and processing. Each pixel in the picture equates to one point in the point cloud. Background points can be filtered out either before or after the processing into point cloud form.

3.1.1.3 Repositioning

After each data capture view is complete it is necessary to reposition the target object to capture another view. Photogrammetry tracking using ORB matching algorithms is employed to automatically rotate the point cloud to match the object rotation Rublee et al. [2011]. In this way the next point cloud can be added to the existing in proper alignment. The process of data capture and repositioning is repeated until the point cloud is completed with no holes from shadowing or lack of views.

3.1.2 Meshing

The scanning process outputs a "point cloud," which is a collection of disconnected points in space that all lie on the surface of the object. However, in order to function, CAD programs require a mathematical description of the surface that those points represent, meaning that the points within the cloud must be joined to their neighbors to show their topographical relationships.

3.1.3 CAD software

Once the STL mesh is created it is possible to create a CAD model from it. Commercially available CAD packages such as SolidWorks have the capability of creating a CAD file from an STL automatically. Depending on the CAD packages algorithm and the complexity of the part there will be a varying level of "intelligence" to the part. This means that the CAD software may have recognized editable features such as holes or extrudes in the model. Once a CAD file is created, the CAD software can be used to heal the geometry of the part in any regions that lacked data, such as the insides of holes. Additionally changes can be made to specific aspects of the part, and new features can be added. Finally, analysis software such as FEA can be used with the CAD model.

3.1.4 Manufacture

Once all changes have been made to the 3D model or mesh it can be used to manufacture new parts. The two simplest ways of manufacturing parts from a 3D scan are using rapid prototyping machines, or CNC machining. CNC machining requires additional creating of tool paths in CAM software before the part to be manufactured. Rapid Prototyping machines on the other hand are capable of creating parts from a mesh file, most commonly an STL.

3.2 Calibration

The techniques outlined in the previous Sections rely on calibration for accurate reconstruction of a single view. Calibration allows the system to measure both intrinsic parameters (such as focal length) and extrinsic parameters (such as relative position and orientation) of the cameras and projector.

3.2.1 Camera calibration

3.2.1.1 Calibration process

Camera calibration consists of determining the intrinsic and extrinsic properties of the cameras [Faugeras, 1993]. The intrinsic properties of the camera are the focal length, image format, principal point, and lens distortions. The focal length is the distance over which initially collimated light is brought into focus. Collimated light refers to light with its rays in parallel.The focal length is an important parameter to triangulation, so even if the manufacturer gives this value it still must be calculated during calibration in order to ensure accuracy. Correcting for lens distortions is critical for obtaining accurate input, and doing so removes distortions from the resulting mesh, vastly improving results. The extrinsic parameters are the location and orientation of the cameras. For simplicity the coordinate system is always defined such that the theoretical pinhole of one of the cameras is at the origin, with the optical axis of that camera aligned with the z-axis of the system. Extrinsic calibration of the cameras allows for stereo vision, which adds to the accuracy of the measurements being taken.

In order to calibrate the cameras it necessary to have a calibration board, which is a flat board or panel with a calibration pattern on it. The calibration pattern is a simple checkerboard pattern with uniform square size. The calibration board is moved to several different orientations, and a picture is captured at each orientation. Figure 3.3 shows the calibration board as seen by both cameras. For intrinsic calibration it is important to have the calibration pattern cover every section of the image throughout the process in order to lens distortion to be characterized. For extrinsic calibration it is essential for both the whole calibration pattern to be captured by both of the cameras in order to properly calibrate for a stereo system. For this reason intrinsic and extrinsic calibration are carried out separately instead of simultaneously as is the

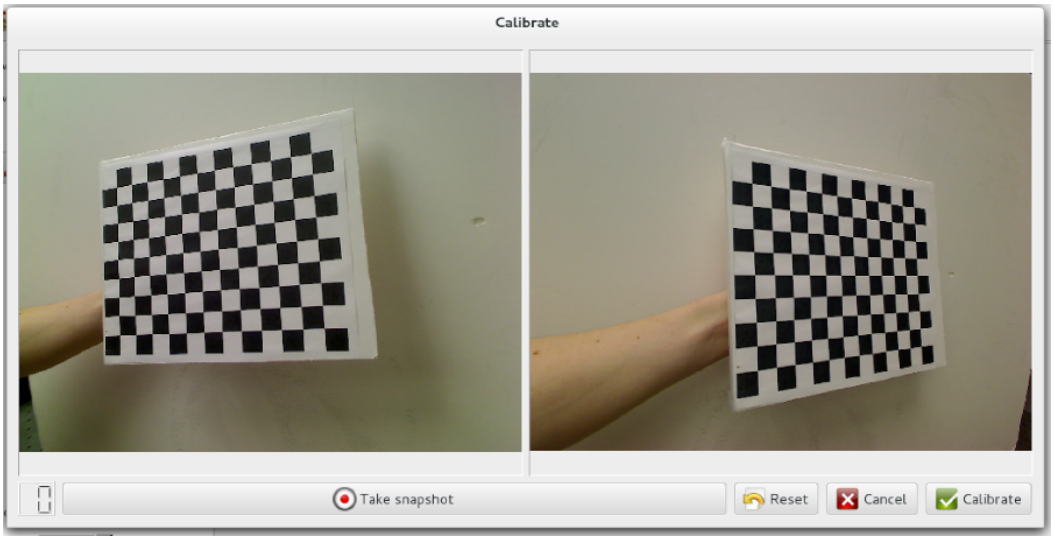


Figure 3.3: The stereo calibration dialog in our software.

norm. First each camera is calibrated for intrinsic parameters, and then the extrinsic calibration is performed for both cameras simultaneously.

3.2.1.2 Calibration algorithms

The technique we used for camera calibration is based on Zhang's method [Zhang, 2000]. This technique relies on observing a plane from multiple orientations. Simply translating the plane does not provide any new information for calculations, and thus is useless for calibration. The method utilizes a corner detection algorithm to find points on the plane. Then the transformations of the points between images in 3D space are calculated. The relationship between the 2D points and 3D points is given by:

$$s \begin{pmatrix} u & v & 1 \end{pmatrix}^T = \begin{pmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{t} \end{pmatrix} \begin{pmatrix} X & Y & Z & 1 \end{pmatrix}, \tag{3.1}$$

where equation s is an arbitrary scale factor, u and v are the coordinates of a point in the camera coordinate system, and (X, Y, Z) is the coordinate of the point in the 3D

coordinate system. The matrix $\begin{pmatrix} \mathbf{R} & \mathbf{t} \end{pmatrix}$ contains the extrinsic parameters, where $\mathbf{R}$ is the rotation matrix relating the two coordinate systems and $\mathbf{t}$ is the translation matrix for the same relation. The remaining variables are the intrinsic parameters, excluding distortion which, due to its nonlinear nature, must be modeled separately; α and β are the scale factors for u and v, respectively; u_0 and v_0 are the coordinates of the principal point of the camera. Finally γ is the skew of the camera, representing a shift from a rectangular pixel array to a parallelogram-shaped one [Zhang, 2000]. For most cameras it is reasonable to assume $\alpha = \beta,$ and that $\gamma = 0$.

From this information it is possible to recover the non-distortion intrinsic and extrinsic parameters of the camera in a closed-form solution. The radial distortion can be calculated using linear least squares. Finally, all parameters are refined to minimize reprojection errors using a Levenberg-Marquardt method [Zhang, 2000]. For more detailed information on the exact solution process please see [Zhang, 2000].

3.2.2 Projector calibration

3.2.2.1 Calibration process

After the camera calibration is complete, the projector calibration can be carried out. Just like with camera calibration, projector calibration consists of determining both intrinsic and extrinsic parameters. The intrinsic parameters are once again focal length and distortion, while the extrinsic parameters are the position and orientation of the projector relative to the coordinate system established during camera calibration.

In order to perform projector calibration the calibration board needs to be in a fixed position where it can be seen by both cameras. Then a series of binary patterns are projected onto the calibration board with the stereo camera system taking an image of each pattern. The patterns consist of first vertical stripes and then horizontal

stripes. These images are then batch processed by the projector calibration algorithm in order to generate the projector calibration parameters. These parameters are used in conjunction with the camera calibration parameters when processing any images taken with the stem into meshes.

3.2.2.2 Calibration algorithms

The method proposed for projector calibration is an existing method that utilizes Zhang's method. First the cameras must be calibrated using Zhang's method as described above. Next the calibration board is placed in a stationary position and its coordinates are calculated using a captured image and the calibration parameters. A checkerboard pattern is projected on top of the existing pattern on the calibration board, and the corner locations are found using the same method as in camera calibration. By calculating the camera rays that pass through the corners of the projection pattern, the location of the corners in 3D space can be found. The projector is treated as the inverse as a camera, and then calibrated through Zhang's method [Falcao et al., 2008].

By the time of the conclusion of the project, projector calibration had not yet been implemented.

3.3 Encoding

Pico Scan, the software we developed, combines two methods to obtain high-quality results. First, the scanner collects low-frequency data via Gray code, also known as reflected binary. Then, the system projects and decodes sinusoidal fringes to collect high-frequency data. This gives the system sub-pixel accuracy with inexpensive equipment.

The software uses a method called *direct coding*, in which every pixel of a camera image is encoded with the coordinates of the projector pixel that originated that ray of light. This is a very flexible approach that can be applied to any setup geometry. The patterns are *temporally multiplexed*, meaning that a single frame is not enough for a measurement; a series of frames is used to compute the result.

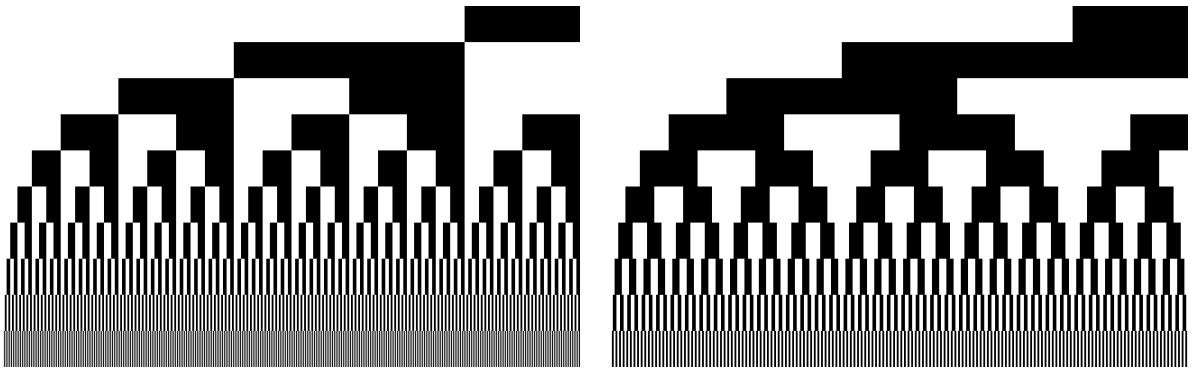
3.3.1 Reflected binary encoding

Binary encoding is a rapid method for data acquisition that can be implemented on inexpensive hardware. First, each projector-pixel index is broken down into the ones and zeros that represent it in software. Next, each bit of the resulting number is projected twice, once as a positive and again as a negative. Finally, for each bit, the two frames are subtracted. A positive difference indicates a binary zero; a negative difference indicates a binary one. These bits are summed for all frames to recover the encoded frame [Scharstein and Szeliski, 2003].

Figure 3.4 displays the progression of a binary structured light sequence. Each row represents a projected frame; time advances as you move down the figure. Notice that in the ordinary binary sequence, a pixel on a bit boundary in a given frame will change between every subsequent frame. The reflected binary sequence does not have this property. See Algo. C.1 and Algo. C.2 for the conversion between binary and reflected binary.

A problem of binary encoding is that pixels on the edge of higher bits are susceptible to noise. This noise can cause the decoded pixel index to vary far from its actual value. In addition, projecting the lowest bit results in a pattern of alternating black-and-white stripes 1 pixel wide. This can cause Moiré issues if the camera and projector are mismatched.

The use of reflected binary solves these issues. A pixel on the edge of a wider binary band, if resolved incorrectly, will only differ from its ideal value by one. This prevents



(a) (b)

Figure 3.4: Progression of: (a) a binary sequence; (b) a reflected-binary sequence.

catastrophic failure during decoding. In addition, the lowest bit of the pattern is a series of stripes 2 pixels wide, reducing the effects of Moiré. These effects may still be present, though, requiring the application of another technique for resolution of finer details.

3.3.2 Phase-shifted fringes

Another method of encoding a pixel's value is through phase shifting. This projects a pattern whose intensity varies sinusoidally from left to right. This pattern is projected multiple times, shifting to the right each time. Decoding this requires a least-squares approach. The foundation for our technique can be found in [Zervas, 2011], Appendix B, and our modification is listed in Appendix.

One issue with phase-shifting is that it is susceptible to noise. Adding more shift steps can reduce the impact of noise but increases the number of patterns required [Zervas, 2011]. In addition, phase shifting requires phase unwrapping, which is computationally intensive and can drastically slow the scanning process. In our implementation, we found that every additional phase shift incurs about 1 second of additional processing time.

3.3.3 Hybrid approach

Our method uses a hybrid binary-encoding/phase-shifting approach. *Pico Scan* uses binary encoding to quickly capture high-level data. Then, it uses phase shifting to enhance this binary data down to sub-pixel levels.

The algorithm proceeds as follows. A number N is selected such that the noise in the pure binary scanning is confined to the smallest N bits; that is, no coded value differs from its ideal value by more than 2^N. This value may be determined by scanning a flat plane and observing the deviations.

Next, the pure binary scan is completed, giving a rough "first pass." This encoding is stored, as well as a "clipped" or "stepped" image in which each value is rounded down to the nearest multiple of 2^N.

The sinusoidal patterns are generated with a period of 2^N projector pixels. The number of shifts may be selected to optimize between speed and accuracy. In our experiences, 24 phase shifts provided an acceptable balance. Future projects can observe the effects of phase shifts and quantize these observations. The fringe images are processed using the techniques outlined in Subsection 3.3.2, generating a phase map with values from $-\pi$ to π. To normalize this phase map into pixel coordinates, 2π is added to all negative values, and every value in the matrix is then multiplied by $\frac{2^N}{2\pi}$.

This scaled phase map gives the position in pixels, accurate to subpixel levels. However, this data repeats itself periodically. To properly perform triangulation, we need to "unwrap" this "wrapped" data. To do this, we simply compute the sum of the scaled phase map and the "stepped" map. This can lead to errors at the edges of steps where the hybrid map differs from the pure binary map by approximately 2^N. To correct this, we simply calculate the difference between the hybrid map and the binary map at every point. If $(hybrid - binary) > 2^N$, we subtract 2^N from that value, and if $(hybrid - binary) < -2^N$, we add 2^N to that value. Due to the periodic

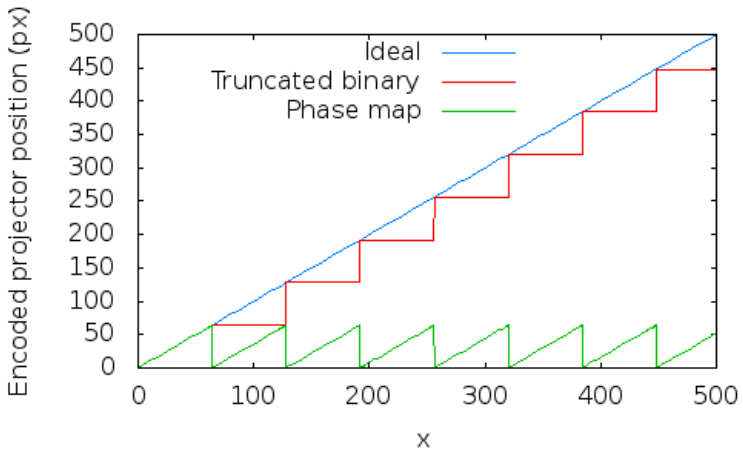


Figure 3.5: A demonstration of the combination of phase data and binary data using our technique.

properties of the sine curve, these cases are the only two that must be adjusted for; larger differences are precluded by the fact that $-\frac{\pi}{2} < \arctan(x) < \frac{\pi}{2}$ for all $x \epsilon \Re$.

For a more visual idea of what's going on, refer to Fig. 3.5. The "Ideal" line represents the ideal result. The binary scan approximates this, but noise is a significant factor in these scans and produces results that are not useful. The "Truncated binary" line represents the noisy binary input, rounded down to the nearest multiple of 2^N. The "Phase map" line represents the highly accurate response of the phase-shifted encoding. Adding the phase map and truncated binary results, in most cases, in the "Ideal" response shown in the graph. In real life, sometimes the phase map is shifted a bit relative to the truncated binary, and so at the edge of each "step" there is a value that is off by roughly 2^N. To correct this, a multiple of 2^N is added or subtracted so that the resulting value is within 2^N of the rough binary value.

3.4 Mesh calculation

Once an encoded image has been captured, this encoding must be translated from (u_c, v_c, u_p) to (x, y, z) coordinates. To do this, we use a simple ray-tracing algorithm,

(a) (b)

Figure 3.6: A sample of: (a) the low-quality binary scan; (b) our high-quality "hybrid" method.

inversely to the way many 3D animation packages generate imagery from a computer file.

3.4.1 Computing central angles of a pixel

Solving the system requires knowledge of θ_c, the angle the incoming ray creates with the camera's optical axis. Figure 3.7 demonstrates the geometry involved in this operation.

By itself, the camera can directly measure neither x_c nor z_c. (If it could, this project would be rather pointless.) However, figure 3.7 demonstrates that

$$\frac{x_c}{z_c} = \frac{x_c^\star}{f_c}, \tag{3.2}$$

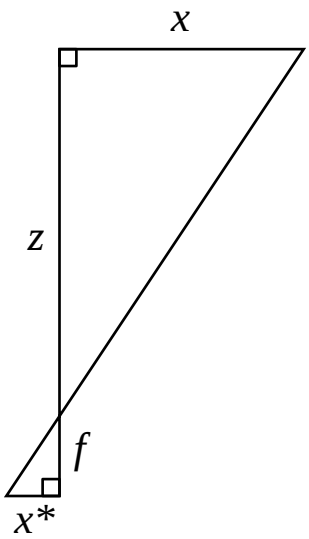


Figure 3.7: Triangulation using focal length and retina resolution.

where f_c is the distance between the focal point and the camera sensor. The sensor is composed of hundreds of pixel sensors, each approximately $2 - 8\,\mu$m across. These sensors give the camera sensor a resolution R_c, measured in $^{\text{mm}}/_{\text{px}}$ or $^{\mu\text{m}}/_{\text{px}}$. These sensors convert the image into pixel coordinates with

$$u'_c = \frac{x^\star_c}{R_c}. \tag{3.3}$$

We can express the camera's scaling factor α_c in terms of pixels with

$$\alpha_c \equiv \frac{f_c}{R_c}. \tag{3.4}$$

By combining Eq. 3.2 with Eq. 3.3 and Eq. 3.4, we get

$$u'_c = \alpha_c \frac{x_c}{z_c}, \tag{3.5}$$

which indicates that, given the normalized focal length, θ_c can be computed with

$$\tan\theta_c = \frac{x_c}{z_c}, \text{ and} \tag{3.6}$$

$$\tan\theta_c = \frac{u'_c}{\alpha_c} \tag{3.7}$$

Similarly,

$$\tan\phi_c = \frac{v'_c}{\beta_c}, \tag{3.8}$$

$$\tan\theta_p = \frac{u'_p}{\alpha_p}, \tag{3.9}$$

$$\tan\phi_p = \frac{v'_p}{\beta_p}, \tag{3.10}$$

and the values u' and v' may be computed with knowledge of the principal point (u_0, v_0) through the relationships

$$u' = u - u_0, \text{ and} \tag{3.11}$$

$$v' = v - v_0. \tag{3.12}$$

Therefore, given any pixel (u, v), we can compute a corresponding pair (θ, ϕ) of angles formed with the optical axis in the x and y planes.

For simplicity, we assume $\alpha_c = \beta_c$ and $\alpha_p = \beta_p$. This is acceptable for most commercially available cameras and projectors, since they are typically manufactured with square pixels [Faugeras, 1993].

3.4.2 Defining a ray through a given pixel

To simplify the explanation of reverse ray-tracing, we define a unit vector function $\hat{\mathbf{X}}$ such that, for a given optical device d at position $\mathbf{P}_d$ and a given pixel (u_d, v_d) from the device, the ray $\mathbf{R}(t) = \mathbf{P}_d + t\hat{\mathbf{X}}(u_d, v_d)$ describes all possible real-world points that

may be imaged or illuminated by that pixel, depending on if the device is a camera or projector, respectively. The definition of $\hat{\mathbf{X}}(u,v)$ is used for both the camera and the projector, as both may be modeled with the pinhole model.

We begin with the definition that $\hat{\mathbf{X}}$ is unit length, and can be defined in local coordinate space using θ and ϕ, as described in Subsection 3.4.1 with

$$\begin{aligned} 1 &= \hat{X}_x^2 + \hat{X}_y^2 + \hat{X}_z^2, && (3.13) \\ \hat{X}_z &= \left[\tan^2\theta + \tan^2\phi + 1\right]^{-\frac{1}{2}}, \text{ and} && (3.14) \\ \hat{\mathbf{X}} &= \begin{pmatrix} \hat{X}_z \tan\theta \\ \hat{X}_z \tan\phi \\ \hat{X}_z \end{pmatrix}. && (3.15) \end{aligned}$$

Since $\tan\theta$ and $\tan\phi$ are themselves functions of u and v, we can define

$$\hat{\mathbf{X}}(u,v) = \begin{pmatrix} \hat{X}_z \frac{u-u_0}{\alpha} \\ \hat{X}_z \frac{v-v_0}{\beta} \\ \hat{X}_z \end{pmatrix}, \tag{3.16}$$

where

$$\hat{X}_z = \left(\left[\frac{u-u_0}{\alpha}\right] + \left[\frac{v-v_0}{\beta}\right]^2 + 1 \right)^{-\frac{1}{2}}. \tag{3.17}$$

The definition of $\hat{\mathbf{X}}$ allows us to compute a ray through any pixel of the camera and, by extension, to form a parametric plane representing a column of pixels from the projector.

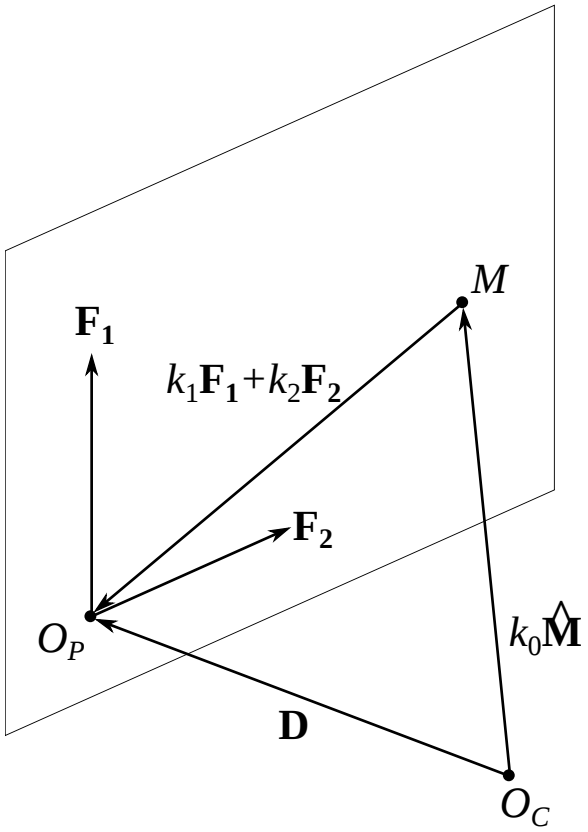


Figure 3.8: Using ray-tracing to locate a point in 3D space.

3.4.3 Intersecting a camera pixel ray with a projection plane

All 3D work in this section is performed using the camera's coordinate system, as shown in Fig. 3.8. We start by assuming a distance vector $\mathbf{D}$ measured from the camera to the projector,

$$\mathbf{D} = \begin{pmatrix} D_x \\ D_y \\ D_z \end{pmatrix}, \tag{3.18}$$

and let $\hat{\mathbf{M}}$ be the unit vector pointing from the camera to the measured point $\mathbf{M}$. $\mathbf{M}$ is measured in the camera's coordinate space.

$$\hat{\mathbf{M}} = \hat{\mathbf{X}}\left(u_c, v_c\right) \tag{3.19}$$

To relate the projector coordinates, we can use the fact that if the projector were to emit a single column of light, there exists a plane in space representing all possible locations that that column could illuminate. Any point in this illuminated plane may be identified uniquely and parametrically as a linear combination of two vectors $\mathbf{F_1}$ and $\mathbf{F_2}$,

$$\mathbf{F}\left(t_1, t_2\right) = t_1\mathbf{F_1} + t_2\mathbf{F_2}. \tag{3.20}$$

Vertical columns, such as we use, will always pass through the projector's local y-axis, meaning that

$$\mathbf{F_1} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}. \tag{3.21}$$

To define $\mathbf{F_2}$, and therefore our plane, we can use the fact that the plane passes through u_p. Since the plane passes through all possible values of v_p, we can select any constant value of v_p to calculate the vector. We arbitrarily choose 0 for this value, and

$$\mathbf{F_2} = \hat{\mathbf{X}}\left(u_p, 0\right). \tag{3.22}$$

Now we have three vectors which sum to $\mathbf{D}$ in a linear combination. We select constants k_0, k_1, and k_2 to solve. Combining this with the projector's arbitrary rotation $\mathbf{R}_p$ and the camera's arbitrary rotation $\mathbf{R}_c$ allows us to solve for triangulation of any arbitrary positioning of camera and projector[1]. $\mathbf{R}_p$ and $\mathbf{R}_c$ are rotation matrices as used in geometric transformations.

We express this linear combination with

$$k_0\mathbf{R}_c\hat{\mathbf{M}} + k_1\mathbf{R}_p\mathbf{F_1} + k_2\mathbf{R}_p\mathbf{F_2} = \mathbf{D}, \tag{3.23}$$

which we solve by computing

$$\begin{pmatrix} \mathbf{R_c}\hat{\mathbf{M}} & \mathbf{R_p}\mathbf{F_1} & \mathbf{R_p}\mathbf{F_2} \end{pmatrix} \begin{pmatrix} k_0 \\ k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} \mathbf{D} \end{pmatrix}, \text{ and} \tag{3.24}$$

$$\begin{pmatrix} \mathbf{R_c}\hat{\mathbf{M}} & \mathbf{R_p}\mathbf{F_1} & \mathbf{R_p}\mathbf{F_2} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{D} \end{pmatrix} = \begin{pmatrix} k_0 \\ k_1 \\ k_2 \end{pmatrix}. \tag{3.25}$$

Now that we have our constants, the point $\mathbf{M}$ may be expressed as

$$\mathbf{M} = k_0\mathbf{R_c}\hat{\mathbf{M}}, \tag{3.26}$$

[1]Note that this does not mean that arbitrary positioning will necessarily yield acceptable results—factors such as device resolution, occlusion, and surface characteristics of the subject mean that the choice of positions is constrained. The significance of arbitrary positioning is that the mathematics is not a constraining factor in selecting a configuration.

with the system origin defined as the origin of the camera's pinhole model. Aligning the global coordinate system with the camera means

$$\mathbf{R}_c = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.27}$$

and Equation 3.26 on the preceding page simplifies to

$$\mathbf{M} = k_0 \hat{\mathbf{M}}. \tag{3.28}$$

3.4.4 Converting to surfels

Applying the above techniques to an input image results in a two-dimensional matrix where each element is a three-dimensional point. This group of points may be immediately exported as a point cloud for quick verification and visualization of results. However, to be most useful in a CAD package, these points must be merged into a mesh. This merging is best done in the scanning software, as the input image implicitly describes relationships between adjacent points.

Our proposed meshing method uses surfels to represent the surface. A surfel, short for *surf*ace *el*ement, represents a small patch of the object's surface in much the same way that a pixel represents a small patch of a computer screen. A surfel stores location, normal, color, and size [Weise et al., 2009]. Figure 3.9 on the next page shows a simple visual representation of a surfel.

There exist many methods and software packages capable of translating surfels into a fully healed mesh. Appendix B on page 59 lists some freely available software libraries and packages that speed this process.

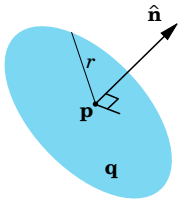


Figure 3.9: A surfel, with location $\mathbf{p}$, normal $\hat{\mathbf{n}}$, color $\mathbf{q}$, and radius r.

3.5 Hardware selection

3.5.1 Projector

The projector was selected to be low-cost, compact, and have a high resolution. The spatial resolution of the scanning system is limited by the resolution of the projector. Additionally the spatial resolution is affected by the distance to the target object, so projectors with a short minimum throw distance were favored. In order to ensure a high spatial resolution at the working distance, the projector also needed to have a large throw ratio. Taking all of these parameters into account, the projector that was selected is the Optoma PK201 pico pocket projector, pictured in Fig. 3.10. See Appendix A for technical specifications of the projector.

Figure 3.10: Optoma PK201 Pico Pocket Projector.

Figure 3.11: Logitech C905 webcam.

3.5.2 Cameras

The cameras for the scanning system were selected to be low cost and high-resolution. The spatial resolution of the cameras needs to be equal to or greater than the spatial resolution of the projector at the working distance. Since cost was a main factor webcams were chosen for the cameras. The specific model selected is the Logitech C905 webcam, pictured in Fig. 3.11. Full specifications on the cameras can be found in Appendix A.

3.5.3 Mounting system

In order to secure the cameras and projector in triangulation geometry a mounting system was created. It was important to be able to adjust the distance between the projector and cameras and the triangulation angle of the system. Additionally, it was critical that the cameras and projector be able to be locked in place so that they would not move relative to each other after calibration.

The mounting system utilized an L-beam as a mounting rail which the cameras and projector could slide along. A custom mount was designed to hold the projector, and the mount was fixed in place on the rail by a hole drilled through the L-beam. The projector mount had a countersunk screw hole for holding the projector in place, and another one for attaching the assembly to a tripod.

Next the existing camera casings were removed so that only the lenses and circuitry remained. New casings were designed that had holes to mount from the bottom. Two

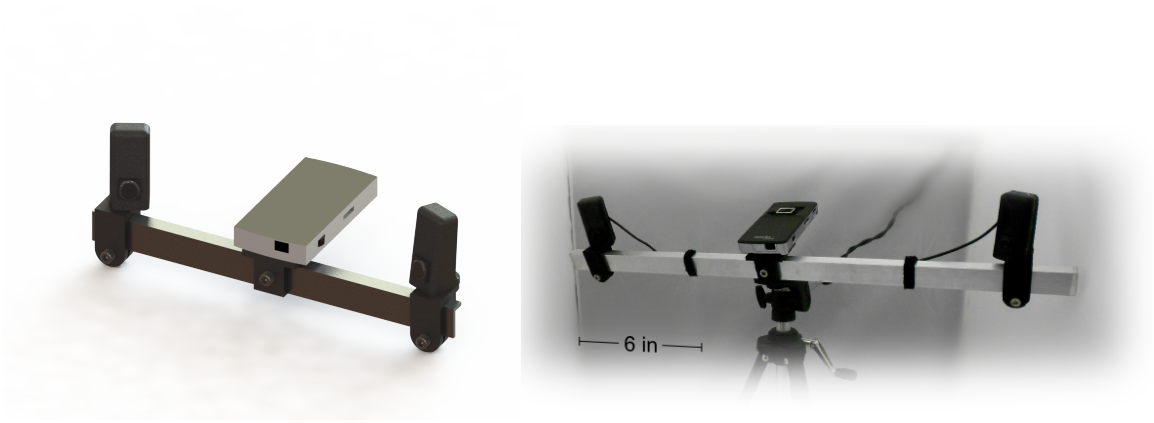


Figure 3.12: Scanner assembly.

mounts were designed to slide on to the L-beam and clamp into place with a screw hole. The mounts had a hole on top for the new camera casings, which could pivot about the hole until tightened into place. The custom mounts and casings were then rapid prototyped. Figure 3.12 shows this setup. Since the highest spatial resolution is achieved at the projector's minimum throw distance, that is the recommended working distance. However, for larger objects a further distance is needed to increase the scan volume. A detailed view of the configuration can be seen in Appendix E, along with detailed CAD drawings of each of the mounting components.

3.5.4 Light box

In order to minimize stray light a light box was purchased for the setup. This minimized the influence of noise on the system and improved results. Figure 3.13 shows the scanning system inside the light box with the front flap open.

3.6 Orientation tracking

To follow the orientation of the object as it rotates between frames, our proposed method is to reconstruct a sparse point cloud using ORB feature tracking, then use least-squares techniques to estimate rotation and translation of this sparse cloud. The

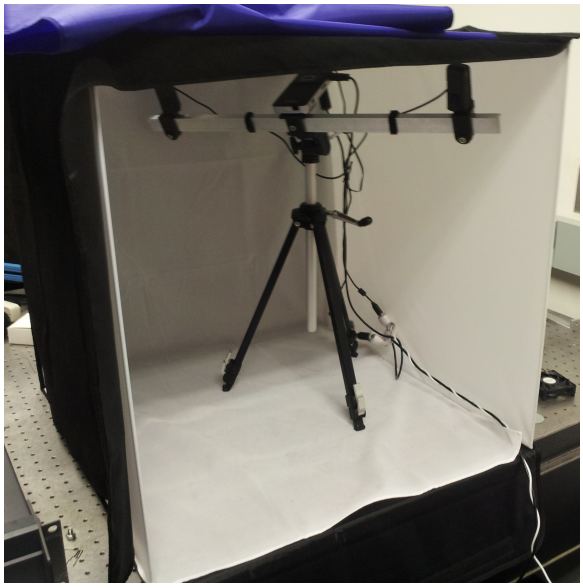


Figure 3.13: Setup in light box.

movement of this cloud matches the movement of the subject, allowing an implementation of this technique to "stitch" multiple single-view scans into a representation of the entire object.

Chapter 4

Results

4.1 Simulation Results

In order to prove the viability of our scanning method, simulated testing was performed. For this testing the software *POV-Ray* was utilized. *POV-Ray* is capable of generating images that simulate the camera and projector in our system. By giving the software the series of patterns to project and the specifications of the camera, projector, and target object, the images can be produced.

For our testing we modeled a theoretically perfect sphere within *POV-Ray*, which had a diameter of 30 centimeters. The center of the sphere was located exactly 40 centimeters from the pinhole of the camera. The modeled camera had a focal length of 8 centimeters, a resolution of 1080p, and a sensor cell size of 7.9 micrometers. After the simulated images were produced, they were analyzed into a point cloud. This point cloud was exported as a Wavefront Object (OBJ) file and further analyzed in *Matlab*. Figure 4.1 contains a downsampled image of the point cloud so that individual points can be distinguished visually.

A *Matlab* script was written to take cross sections of data from various angles. All of the cross sections passed through the center of the sphere, so theoretically they

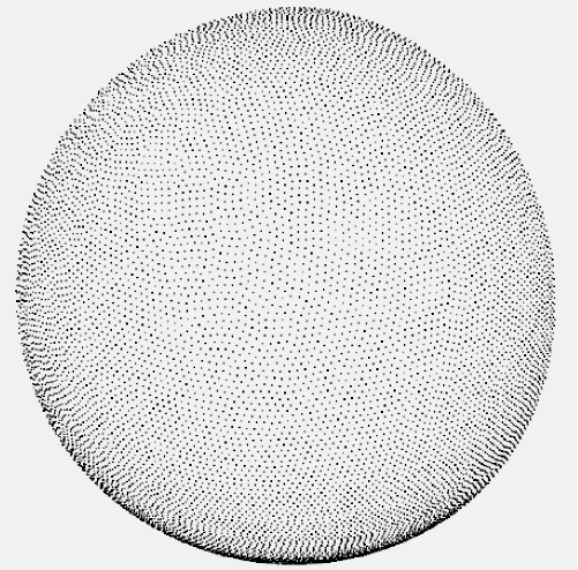


Figure 4.1: Downsampled image of point cloud calculated from simulation data.

should all have a radius of 15 centimeters. Cross sections were taken by rotating the sphere around each of the three major axis, at an interval of every 5 degrees. Since the point cloud is composed of discrete points the cross sections needed to have a tolerance of $\frac{1}{2}$ the spatial resolution in order to ensure data points would fall in the cross section. An existing *Matlab* function was incorporated into the script to calculate the radii of the cross sections. Additionally the standard deviation of the data points within each cross section was calculated. Figure 4.2 shows the calculated radii, while Fig. 4.3 shows the standard deviations.

One of the first pieces of information this data reveals is that the results are worse near the edges of the scan. This is expected and there are two reasons for it. From the camera's perspective the edges of the scan appear to be at nearly a 0 degree angle, greatly reducing the accuracy of the triangulation. To compensate for this, a quality map was used to remove extraneous points—if illumination of a point did not increase its brightness by a specified amount, then that point was assumed to be noise. However, this reduced the point cloud density at the edges, which in turn reduced the sampling size for the cross sections. This in turn produced inaccurate radii for those

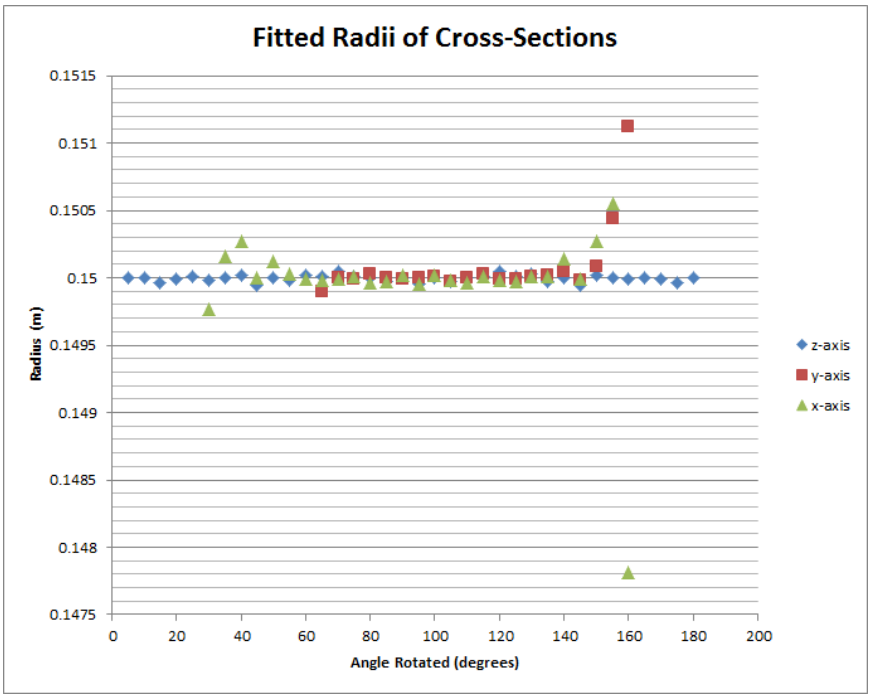


Figure 4.2: Cross sectional radii of Fig. 4.1 showing erroneous data near the edges.

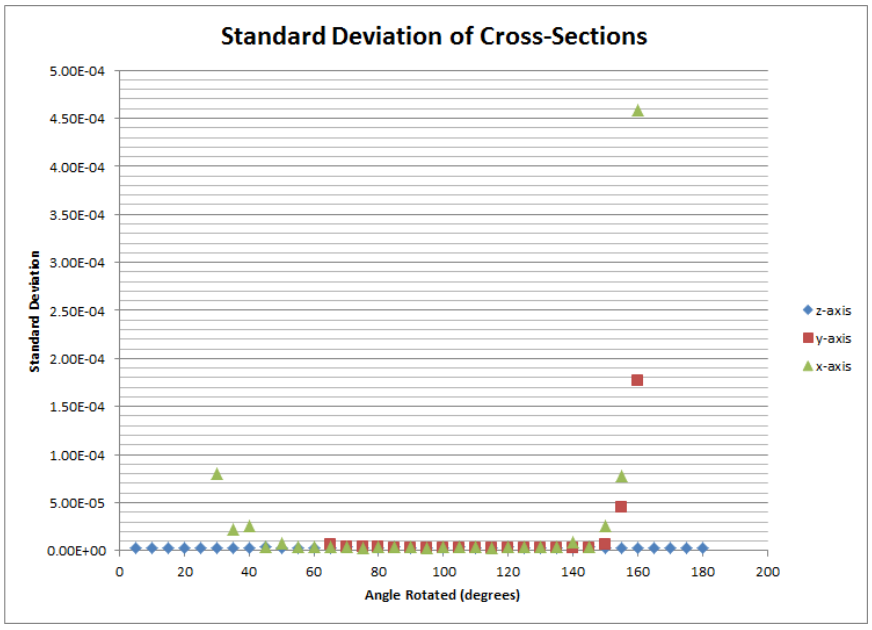


Figure 4.3: Standard deviations of Fig. 2 showing high deviation near edges.

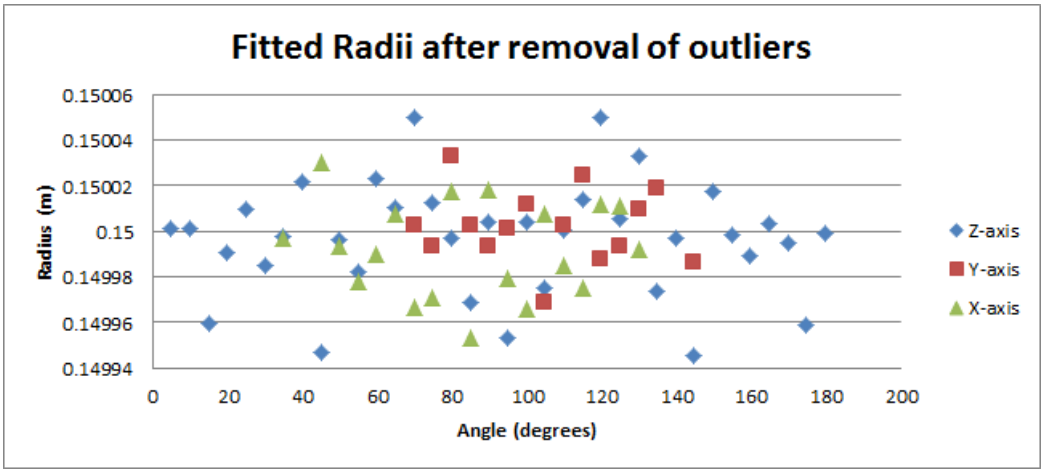


Figure 4.4: Radii after outlier removal.

Table 4.1: Simulated results.

Data Set	Mean Radius (m)	Standard Deviation (µm)	Error (µm)
x	0.1499920	21.1	-7.9
y	0.1500019	16.1	1.9
z	0.1499963	25.0	-3.6
All	0.1499968	20.7	-3.2

cross sections. In order to more accurately analyze the data, the erroneous points were removed using Chauvenet's criterion [Taylor, 1997]. Figure 4.4 is an updated graph with the outliers removed.

This chart shows what appears to be a random scattering of data points within a small range about the expected radius. There is no clear directionality to the quality of the results, which indicates that the method is equally accurate in all directions. Further analysis was done upon these results to find the average radius and standard deviation for each rotation direction, and for the whole data set. Table 4.1 contains this analysis.

This reveals that the system has an overall accuracy of around ±3.2 micrometers and a precision of ±20.7 micrometers. These results are suitable for many engineering applications, and thus the method has merit to continue developing. It should be noted that no decimals were dropped for significant figures in these calculations because the limitations of the measuring method were unknown. In fact these cal-

culations were preliminary testing to find those very limitations. Additionally these results are with a single simulated camera rather than a stereo system, and with only the binary pattern being projected.

4.2 *Pico Scan*

Pico Scan is the custom software created for this project. *Pico Scan* was programmed in C++ and has a graphical user interface (GUI) that is built in the Qt framework. *Pico Scan* can be used to run simulations or perform scans.

4.2.1 Calibration

Pico Scan has a built in camera and projector calibration process that utilizes the calibration algorithms detailed in the calibration section. While calibrating the cameras the real time images from both cameras is displayed in the program. Once calibration is complete the calibration results are automatically stored and used in any subsequent scans until they are either cleared or overwritten with a new calibration.

4.2.2 Scanning

Pico Scan creates a single view scan of an object with a single button click. Once the scan button is pressed the software automatically projects the binary patterns and saves images from the cameras of each pattern. Additionally it is possible to chose how many images are taken and averaged together for each pattern. Once the images are all taken they are automatically processed by the triangulation algorithm and a point cloud is generated.

In addition to generating point clouds directly from a scan, *Pico Scan* is capable of generating point clouds from two other sources. The first is from a series of image files captured externally. The image files must contain the same reflected binary patterns,

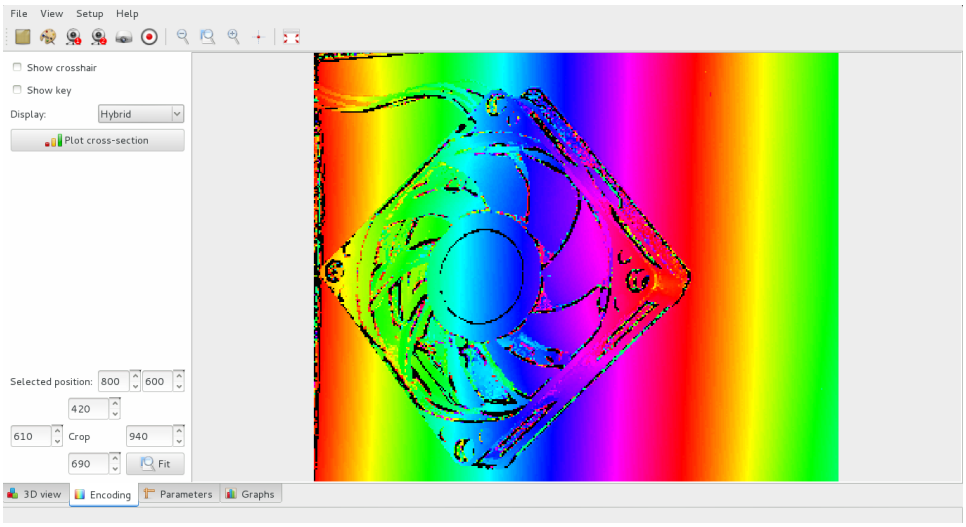


Figure 4.5: A screenshot of the "Encoding" view of *Pico Scan*. This displays the various coded frames and allows the user to create plots similar to Fig. 4.7 directly in the software. The data shown is the data used to generate the mesh in Fig. 4.6.

and the filenames must be formatted correctly for *Pico Scan* to process them. The other source *Pico Scan* can process images from is simulations. Certain versions of *Pico Scan* are capable of interfacing with the *POV-Ray* simulation software to generate images of an object with the same projected patterns. These images are then processed in the same way as real scan data.

4.2.3 Encoding view

Pico Scan allows the user to inspect the coding data after decoding has occurred but before the 3D meshing process takes place. Figure 4.5 shows the "Encoding" view after a scan. The view may be cropped for close inspection of a part of the coded image. The color mapping maps pixel value to hue by the equation:

$$\text{hue} = \frac{\text{encoded value} \times 360^\circ}{1024} \tag{4.1}$$

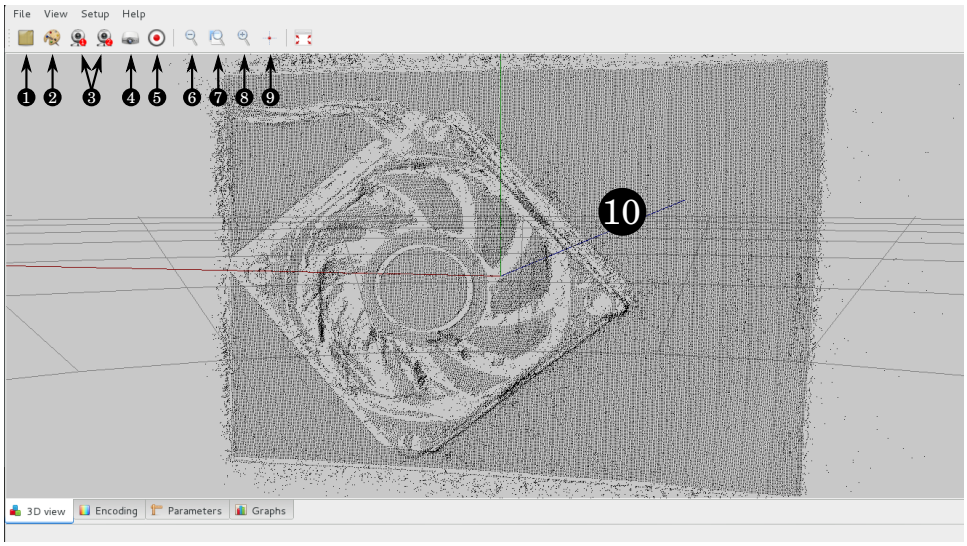


Figure 4.6: A screenshot of *Pico Scan*'s 3D view with data loaded. (1) select calibration pattern; (2) define a "background", so it may be removed from subsequent scans; (3) calibrate each camera; (4) calibrate stereo and projector (unfinished); (5) take 3D frame; (6) zoom in; (7) reset zoom; (8) zoom out; (9) center mesh in virtual world; (10) mesh view.

This is for a quick visual inspection only, to demonstrate the difference between the rough binary data and the smooth hybrid data. For more quantitative results, the user may plot a cross-section directly from the software.

4.2.4 Object view

Pico Scan renders point cloud data in a virtual three-dimensional environment. The point cloud can be viewed from any angle as the coordinate system is rotated. Additionally wire-frame models of the cameras and projector are displayed in this view and are located at the positions indicated by the extrinsic calibration. Object view utilizes OpenGL, which is an open source library for rendering 2D and 3D graphics. Refer to Fig. 4.6 and for screenshots of the program.

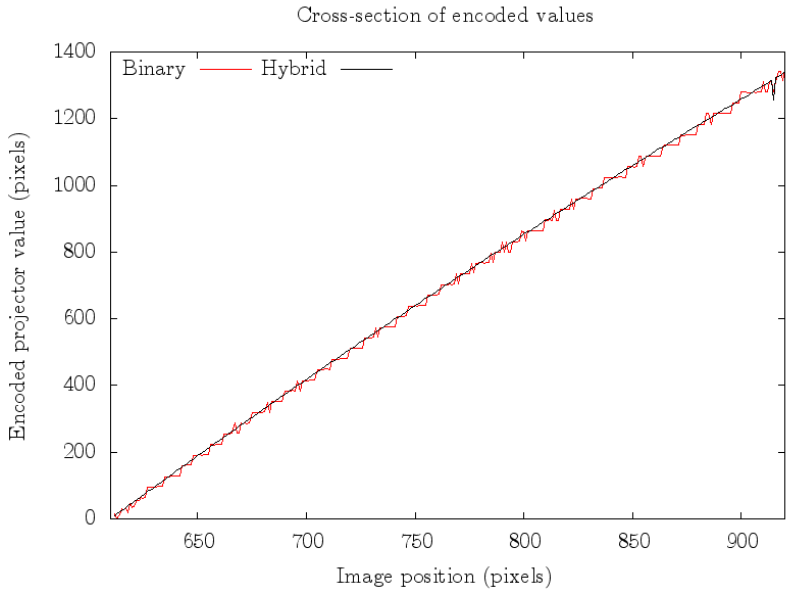


Figure 4.7: Comparison of quality of binary vs. hybrid method. This scan is of a flat surface, and the ideal response would be a straight line. The binary data exhibits noise, while the hybrid data exhibits a smooth, useable scan.

4.3 Scanning results

Even from a strictly qualitative perspective, the hybrid method offers clearly superior results over the binary method. Figure 4.7 displays a pair of encodings. The "Binary" line was generated using the pure binary method; the "Hybrid" line was generated using the hybrid method. Note how much smoother the hybrid data is than the binary data.

This high quality translates well to the generated meshes. Figure 4.8 demonstrates the remarkable improvements that the hybrid technique offers. The left scan was generated using only the binary technique, while the right scan used the initial rough scan to unwrap the sinusoidal data. Notice in particular the hair; on the left it is barely discernible, while on the right it is easily distinguished.

(a) (b) (c)

Figure 4.8: Comparison of scanning methods. Scan (a) was performed with only the binary technique; scan (b) used binary-unwrapped fringe projection. This demonstrates qualitatively the superiority of our developed hybrid method. Image (c) shows the original statue.

Currently, the meshes require a great deal of post-processing before they can be printed or used in a CAD program. Figure 4.9 shows examples of the system's capabilities.

(a) (b) (c)

Figure 4.9: Sample meshes generated from *Pico Scan* and post-processed, along with photos of the objects from which they were generated. (a) WPI's "Proud Goat"; (b) a decorative statue; (c) a section of a broken component.

Chapter 5

Conclusions

This project set out to create a low cost system to digitize existing objects in full 3D for the purpose of enhancing the engineering design cycle. At the end the results we achieved showed that this is indeed possible with commercially available technology and that with more time these goals could have been fully satisfied. The hardware that was selected met the needs of the system and were sufficient to achieve engineering quality scans. Additionally the hybrid method that was proposed produced results that met the goals of the project. The only holdup was with the software development. The entire framework has been set up to complete the goal of accurate, precise, and fully three-dimensional scans, but the software implementation is as of yet incomplete. With the current state projector calibration is currently incomplete preventing perfectly accurate scans. Additionally this prevented the implementation of the photogrammetry tracking for automated stitching of the meshes. For this reason it is recommended that future iterations of this project contain at least one computer science major to facilitate the implementation of software improvements.

Despite the fact that the software is not yet capable to realizing the full goal of the project, the theory is sound. The project was still a success because it proved that low cost hardware was in fact capable of achieving measurements that could be

used for engineering design or analysis. Additionally the current system is capable of producing the desired results, just not in an automated manner. Through careful measurements of the projector location accurate scans can still be achieved without the calibration being fully functional. Cleaning of the stray noise can be done manually, and with great effort multiple views can be aligned and stitched. The work left to be done will only serve to further improve and validate this already functional system.

5.1 Future Work

The entire scanning system was researched and designed from start to finish, however there are many components that were not implemented due to time constraints. Future MQPs can pick up where we left off and finish these components. The projector calibration is currently incomplete, thus current scans use calibrated cameras and manually measured projector parameters. Furthermore stereo vision requires precise projector calibration to be implemented properly. The reason for this is that the two point clouds from the separate cameras will be distorted differently from the inaccurate projector position depending on their relative locations. Stereo vision will greatly improve results by automatically removing noise data that is only seen by one of the cameras. Additionally it will allow for averaging of the actual shape data in regions that both cameras can see. Photogrammetry tracking also requires the projector calibration in order to function properly. The addition of photogrammetry tracking will allow for the automated alignment and stitching of meshes, and thus is essential for creating full 3D CAD files in a timely manner. Additional measures to automatically remove noise such as a "nearest neighbor" filter should be implemented to automatically clean up the extraneous points caused by stray light. The meshing algorithm needs to be improved, perhaps through the use of the open source library

Point Cloud Library (PCL). Finally, the whole program needs to be optimized for speed and processing efficiency. Specifically multi-threading needs to be implemented so that the program can perform multiple tasks at once, and the actual data acquisition process needs to be speed up through improvements in the camera-projector synchronization.

In addition to the incomplete components of the project future work needs to be done in order to characterize the quality of results obtained by the system. Detailed analysis on the recovered images should be performed on several different standardized gauges in order to determine the accuracy and precision of the system. These analyses should include cross section plots of the shape information and comparisons of the quality of the binary and hybrid methods, both with and without stereo cameras.

Appendix A

Equipment used

A.1 Camera

The camera is a Logitech C905 Webcam. Specifications are listed in table A.1.

Table A.1: Camera properties.

Resolution	1600 x 1200
Frame rate	Up to 30 fps
Bit depth	Up to 8 bits/pixel
Min Focal Length	50 mm
Sensor cell size	2.8µm x 2.8µm

A.2 Projector

The projector is an Optoma PK201 Pico Pocket Projector. Specifications are listed in table A.2.

Table A.2: Projector properties [Optoma].

Projection Type	DLP
Native Resolution	854 x 480
Max Resolution	1280 x 800
Brightness	20 ANSI Lumens
Contrast Ratio	2000:1
Light Source	LED
Throw Ratio	2.2:1
Projection Distance	10” -126”
Weight	0.35lb
Dimensions	2.4”x0.7”x4.6”

Appendix B

Open-source libraries and programs

Pico Scan was made possible by the many open-source projects available today. Using these projects meant we didn't have to invest time writing, for instance, matrix multiplication algorithms or camera drivers, allowing us to more quickly develop our novel method for digitization. This appendix lists the libraries and programs we used in this project. All are freely available for all operating systems.

Pico Scan was initially written and run on a laptop running Ubuntu 12.10 "Quantal Quetzal".

B.1 *OpenCV*

OpenCV is an open source computer vision library created by Intel in 1999 for the purpose of advancing the computer vision field. *OpenCV* has a BSD software license, which is a minimally restricting license. *OpenCV* source code is free to use for academic and commercial purposes so long as the proper acknowledgments and copyrights are used. The *OpenCV* library contains a vast amount of source code

for various computer vision application, and is supported by a large open source community [itseez, 2013].

OpenCV code was used in this project as part of the scanning program. The main use of *OpenCV* is in the calibration software. The camera calibration code for locating the corners on a checkerboard pattern was created using *OpenCV* source code. The other optional calibration patterns supported by the software also use *OpenCV* code. Additionally the solvePNP code used to match features for stereo calibration also utilizes *OpenCV*. The use of existing *OpenCV* source code to perform these complicated and vital tasks allowed for time to be focused on other areas of the development. This of course freed up vital time and allowed for other areas of the software to be more robust. The *OpenCV* stereo calibration code was also modified and re-purposed for the projector calibration. In this way the *OpenCV* library is responsible for most of the calibration code. Additionally, *OpenCV* code is used for the photogrammetry tracking used to track the orientation of the object being measured. In this way the *OpenCV* libraries form a crucial backbone for our software, working in support of our triangulation algorithm and image processing code.

The *OpenCV* community maintains a homepage at `http://opencv.org`.

B.2 *Point Cloud Library*

Point Cloud Library (PCL) is a library for working with raw acquired 3D data. It allows users to align multiple scans, form meshes from point clouds, and smooth results. We planned to use PCL to prepare our data for export, but there was not enough time left after implementation of the hybrid technique to integrate PCL.

More information on PCL is available at `http://pointclouds.org`.

B.3 *OpenGL*

OpenGL is an open source library for developing 2D and 3D graphical applications. It was created in 1992 and has become the industry standard for graphical programming. It is used in a variety of applications from animation to CAD software [The Khronos Group, 2013].

In this project *OpenGL* was used to create the 3D viewing environment for the point cloud data. This environment includes a coordinate system with the three major axes visible, as well as wire-frame models of the cameras and projector, located in their positions and orientations. When calibration is complete, these positions and orientations may be updated to match the new data.

B.4 *Qt* and *Qt Creator*

Qt is an extensive framework for object-to-object communication and user interface construction. It was used for the user interface and workflow management in our software. *Qt Creator* is a development environment designed for building *Qt* projects. For more information, visit `http://qt-project.org`.

B.5 *gnuplot*

Pico Scan is able to interface with the *gnuplot* program if it is installed on the user's system in order to plot cross-sectional data obtained through scans. If *gnuplot* is not installed, *Pico Scan* will work just fine, with the exception of an inability to produce graphs of this data. Documentation and downloads for *gnuplot* are available at `http://www.gnuplot.info`.

B.6 *MeshLab* and *Blender*

MeshLab and *Blender* are two programs used for editing meshes. *MeshLab* is useful for initial cleaning of data, filling of small holes, and aligning multiple scans with a relatively automatic process. *Blender* is useful for manual cleaning and alignment, as well as for producing modified versions of meshes. Both were used to post-process meshes generated by *Pico Scan*. *MeshLab* and *Blender* are available at `http://meshlab.sourceforge.net` and `http://blender.org`, respectively.

B.7 *POV-Ray*

POV-Ray is a ray-tracing program that allows the user to write scripts that simulate scenes. This greatly enhanced our ability to quickly simulate different hardware capabilities, and an early version of *Pico Scan* used *POV-Ray* to test the validity of the binary method.

Appendix C

Algorithms

C.1 Reflected binary pattern generation

These four functions describe how the binary projection patterns are encoded and decoded, from binary to reflected binary and back. Algorithm C.1 converts ordinary binary into reflected binary, while Algorithm C.2 performs the inverse operation. Algorithm C.3 demonstrates how to generate projection patterns for the ordinary binary sequence. Algorithm C.4 describes how to use the encoding and decoding functions to generate the improved reflected-binary projection patterns, and Algorithm C.5 describes how to process a reflected binary-coded image into an ordinary binary-coded image. Using the reflected-binary method, as described in Subsection 3.3.1 on page 28, offers several advantages over the ordinary binary format.

Algorithm C.1 Algorithm in C for converting ordinary binary to reflected binary.

```
int binaryToGray(int num) {
        return (num >> 1) ^ num;
}
```

Algorithm C.2 Algorithm in C for converting reflected binary to ordinary binary.

```
int grayToBinary(int num)
{
    int numBits = 8 * sizeof(num);
    int shift;
    for (shift = 1; shift < numBits; shift = 2 * shift)
    {
        num = num ^ (num >> shift);
    }
    return num;
}
```

Algorithm C.3 Pseudocode algorithm for generating sequential ordinary binary-encoded patterns. The | and $\ll$ operators represent the standard bitwise-or and bit-shift operators, respectively, as used in the C programming language.

function *createOrdinaryBinaryPattern* (*rows*, *columns*, *bit*):

1. create a $rows \times columns$ matrix P
2. loop u from 1 to *columns*:
   (a) $orValue = u \mid (1 \ll bit)$
   (b) if $(orValue = 0)$ then $binaryValue = 0$
   (c) otherwise $binaryValue = 1$
   (d) loop v from 1 to *rows*:
      i. $P_{vu} = binaryValue$
3. return P

Algorithm C.4 Pseudocode algorithm for generating sequential reflected binary-encoded patterns. The | and $\ll$ operators represent the standard bitwise-or and bit-shift operators, respectively, as used in the C programming language.

function $createReflectedBinaryPattern\,(rows,\ columns,\ bit)$:

1. create a $rows \times columns$ matrix P
2. loop u from 1 to $columns$:
   (a) $reflectedBinary = binaryToGray\,(u)$
   (b) $orValue = reflectedBinary\ |\ (1 \ll bit)$
   (c) if $(orValue = 0)$ then $binaryValue = 0$
   (d) otherwise $binaryValue = 1$
   (e) loop v from 1 to $rows$:
      i. $P_{vu} = binaryValue$
3. return P

Algorithm C.5 Pseudocode algorithm for converting a reflected binary-coded image to an ordinary binary-coded image. Argument **I** is a $rows \times columns$ matrix containing encoded information.

function $decodeCameraImage\,(\mathbf{I})$:

1. create an output matrix **E** with the same dimensions as **I**
2. loop u from 1 to $columns$:
   (a) loop v from 1 to $rows$:
      i. $E_{vu} = grayToBinary\,(I_{vu})$
3. return **E**

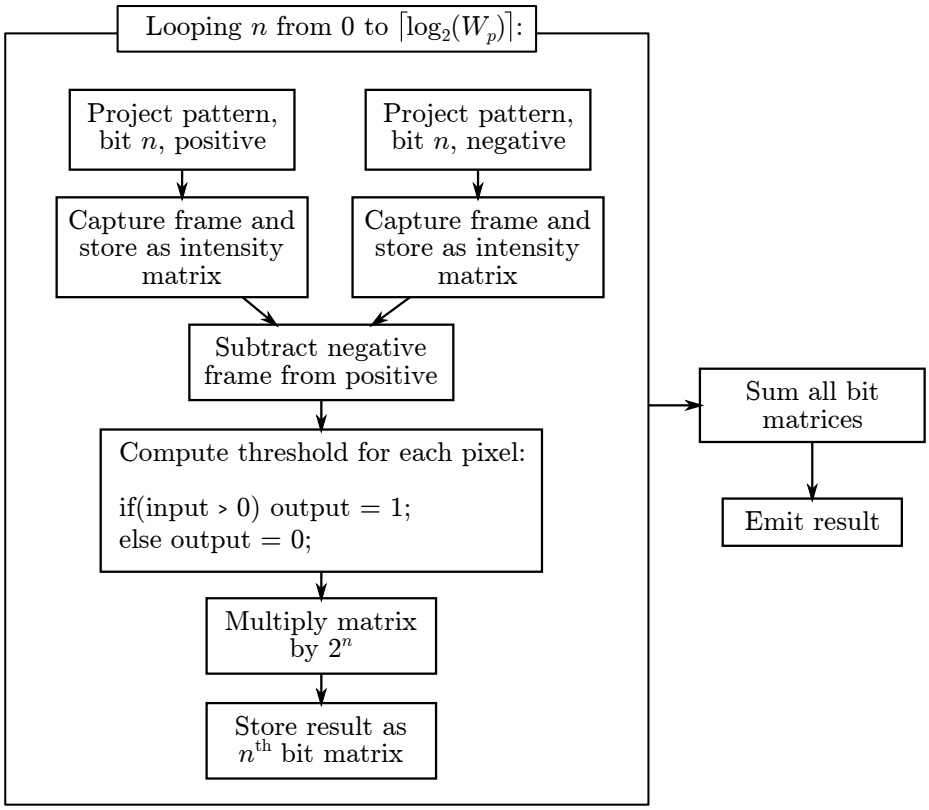


Figure C.1: Diagram of binary capture method. This work flow generates results for both the ordinary and reflected binary techniques, depending on the patterns selected for projection.

C.2 Binary capture method

Figure C.1 demonstrates how the algorithms, image capture, and output are correlated to produce a single binary-coded frame in a form suitable for triangulation.

C.3 Phase recovery

Algorithm describes the method we use to convert a series of intensity maps to a single scaled phase map, where each entry in the matrix represents $u_p \mod 2^N$. Argument **L** is a list of intensity maps, each phase-shifted by an equal amount from the previous and covering the entire 2π period of the sine wave. This algorithm uses

the two-argument form of arctan to avoid discontinuities at 2π intervals. In C this function is implemented as `atan2(y, x)`. Argument Q is a quality threshold, and only computes phase for elements where peak-to-peak sinusoid magnitude is greater than Q in the camera's intensity units.

Algorithm C.6 Pseudocode algorithm for converting a series of intensity maps to a scaled phase mapping.

function $computeScaledPhaseMap\,(\mathbf{L}, N, Q)$:

1. If the elements of $\mathbf{L}$ are not all the same size, throw an error.
2. Let $\alpha = \frac{2\pi}{length(\mathbf{L})}$.
   - Note: the use of α here is different from its use in camera calibration. The two are not related.
3. Let $\mathbf{P}$ be a matrix with dimensions the same as each element of $\mathbf{L}$.
4. Let $\mathbf{A} = \sum_{n=0}^{length(\mathbf{L})-1} \begin{pmatrix} 1 & \cos(\alpha n) & \sin(\alpha n) \\ \cos(\alpha n) & \cos^2(\alpha n) & \cos(\alpha n)\sin(\alpha n) \\ \sin(\alpha n) & \cos(\alpha n)\sin(\alpha n) & \sin^2(\alpha n) \end{pmatrix}$
5. Looping through each (u, v) in $\mathbf{I}$:
   (a) Let $\mathbf{b} = \sum_{n=0}^{length(\mathbf{L})-1} \begin{pmatrix} \mathbf{L}_{vu} \\ \mathbf{L}_{vu}\cos(\alpha n) \\ \mathbf{L}_{vu}\sin(\alpha n) \end{pmatrix}$.
   (b) Let $\mathbf{X} = \mathbf{A}^{-1}\mathbf{b}$.
   (c) If $X_1 \geq Q$ (if the magnitude of intensity fluctuation is greater than the desired threshold)
      i. Let $\mathbf{P}_{vu} = \frac{\arctan(X_3, X_2)}{2\pi}$.
      ii. If $\mathbf{P}_{vu} < 0$, let $\mathbf{P}_{vu} = \mathbf{P}_{vu} + 1$.
   (d) Otherwise, let $\mathbf{P}_{vu} = \varnothing$ (null entry).
      - Note: in reality, the null entry is represented by a negative value, and negative values are simply ignored during triangulation.
6. Return $\mathbf{P}$, the scaled phase mapping.

Appendix D

Using *Pico Scan*

This section outlines how to generate a scan using *Pico Scan*.

D.1 Setting up the hardware

The hardware, as designed, is fairly simple to set up. The cameras and projector need to be facing the same way on the L-beam. Use the adjustment screws to move the assembly, then tighten them to prevent movement after calibration.

If you are using *Pico Scan* with your own hardware, ensure you can set it up such that the cameras and projector are stationary relative to each other. Each camera should be about 20 cm from either side of the projector, with the optical axes of the cameras angled about 30° to that of the projector.

D.2 Installing the software

Pico Scan is currently distributed in source form to allow for the widest possible audience. It should work on Windows, Mac OS X, and Linux.

D.2.1 Installing dependencies

Pico Scan depends on several libraries and programs to compile and run properly. You should install them in this order:

1. *Qt Creator* (see Section B.4 on page 61 for details)
2. *OpenCV* (see Section B.1 on page 59 for details)
3. *gnuplot* (optional; see Section B.5 on page 61 for details)

Once the dependencies are installed, open the *Pico Scan* project file in *Qt Creator* and click Build. Your computer should compile the source into an executable.

D.2.2 Setting parameters manually

Since extrinsic calibration has not been completed, parameters must be set manually. Open `mainwindow.cpp` and edit the parameters listed in the constructor in order to set extrinsic parameters.

D.3 Running *Pico Scan*

To run *Pico Scan*, first make sure that the projector has been connected to your computer as an external monitor. Make sure that your operating system recognizes the projector as the "secondary" display and not the "primary" one, as this will prevent the scanner from functioning. Build the project again and click Run.

Using the program is fairly straightforward.

D.3.1 Calibrating the cameras

First, if you wish, you can calibrate the two cameras. Currently, however, the program only uses one of these cameras to reconstruct the mesh. To do this, click on

the “Camera 1” or “Camera 2” buttons on the toolbar. Click the “take snapshot” button when the calibration standard is in view. The program should take a couple of seconds to locate the pattern and add the points to memory. If the pattern-finding was successful, the counter in the bottom-left corner will increment. Reposition the chessboard and take more snapshots. Typically, the calibration converges acceptably after about 12 frames.

D.3.2 Setting parameters

Click the “parameters” tab to set scanning parameters. The only parameter you can reliably change is the number of phase shifts. For acceptable results in a reasonable amount of time, we found that 24 shifts produced good results.

D.3.3 Collecting data

Next, place the subject in the area illuminated by the projector. Orient it so that the camera can see as much of the object illumination as possible. (You can use the camera calibration dialog to verify this.) Press the red “Take frame” button or press `Ctrl` + `R` on the keyboard to run the scanning sequence. The patterns will project in sequence and after they’ve finished processing your mesh will appear in the 3D view.

D.3.4 Analyzing and exporting data

To save your mesh in the STL format, click File → Export to STL. Navigate to the folder where you wish to store the file and click Save.

To view the encodings, click on the Encodings tab. This allows you to crop the coded views and see a color mapping of the coded values. You can select between Binary, Phase mapped, or Hybrid views of the data. By typing a number into the

second "Selected position" box, you can indicate a y-value that will then be used as the basis to plot a cross-section (assuming you have installed *gnuplot*).

Appendix E

CAD Drawings

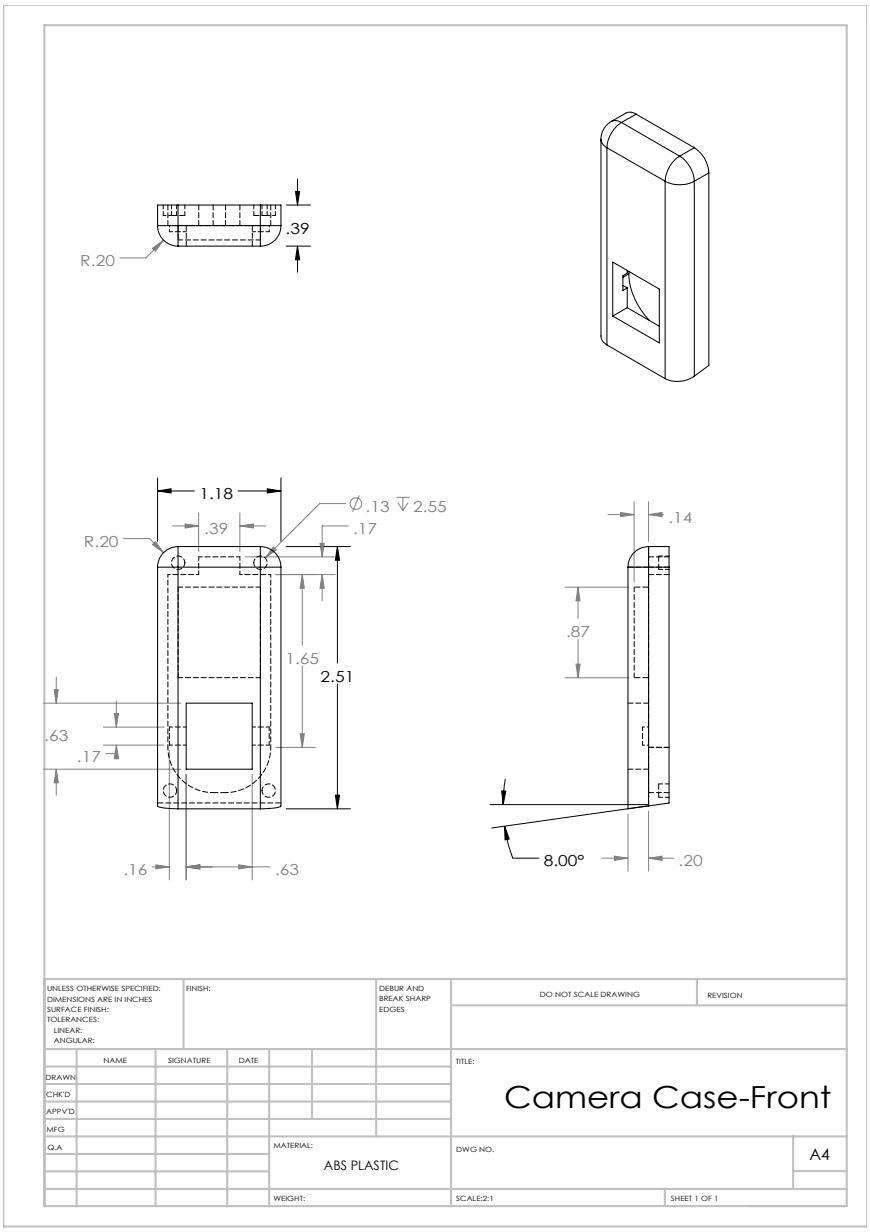


Figure E.1: Camera case—front.

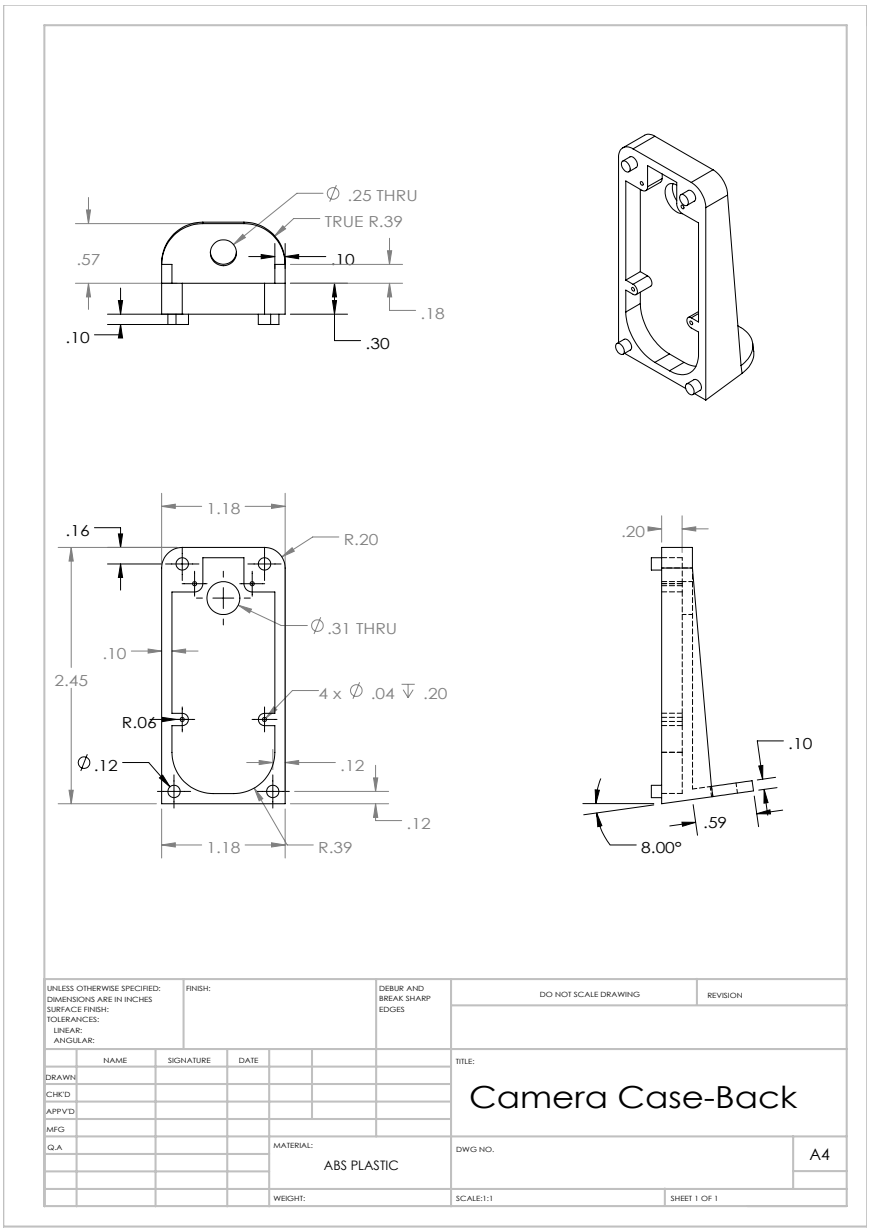


Figure E.2: Camera case—back.

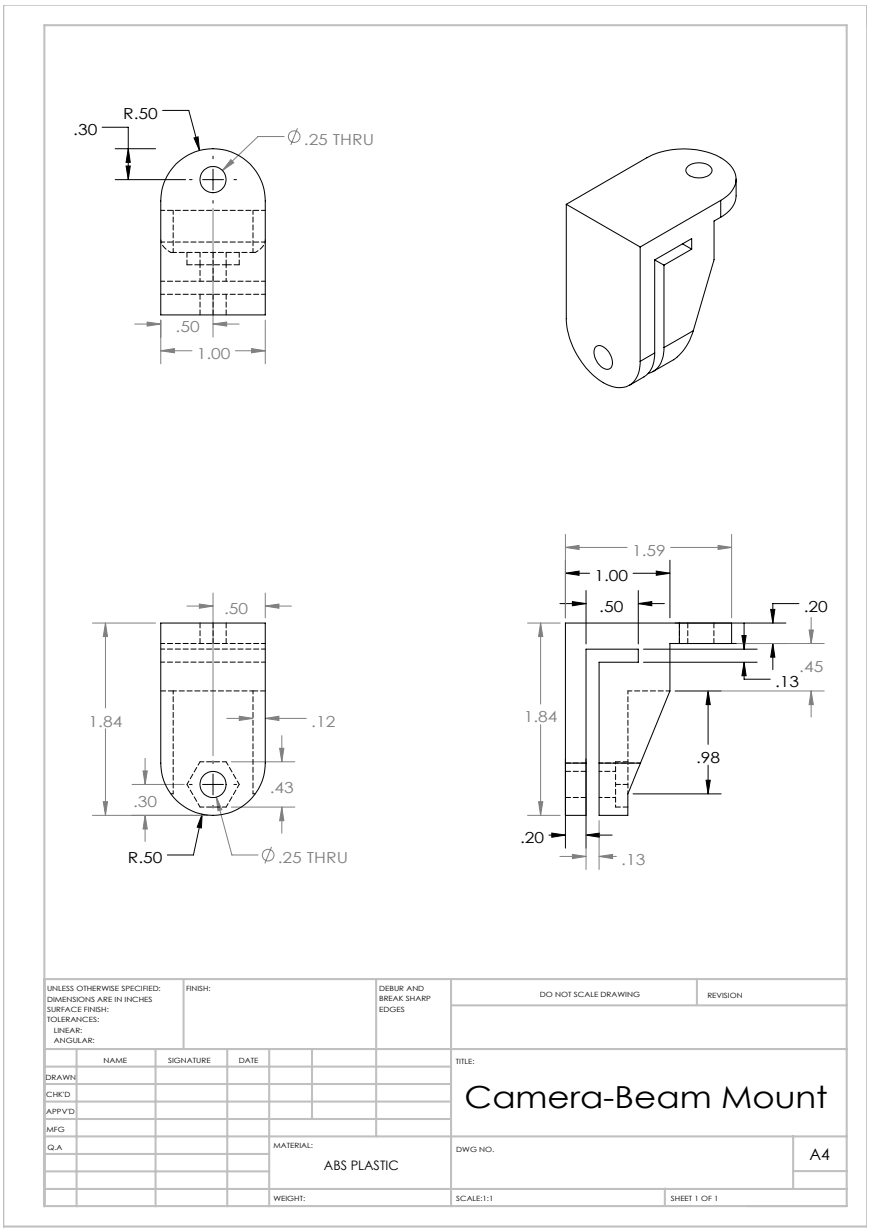


Figure E.3: Camera—beam mount.

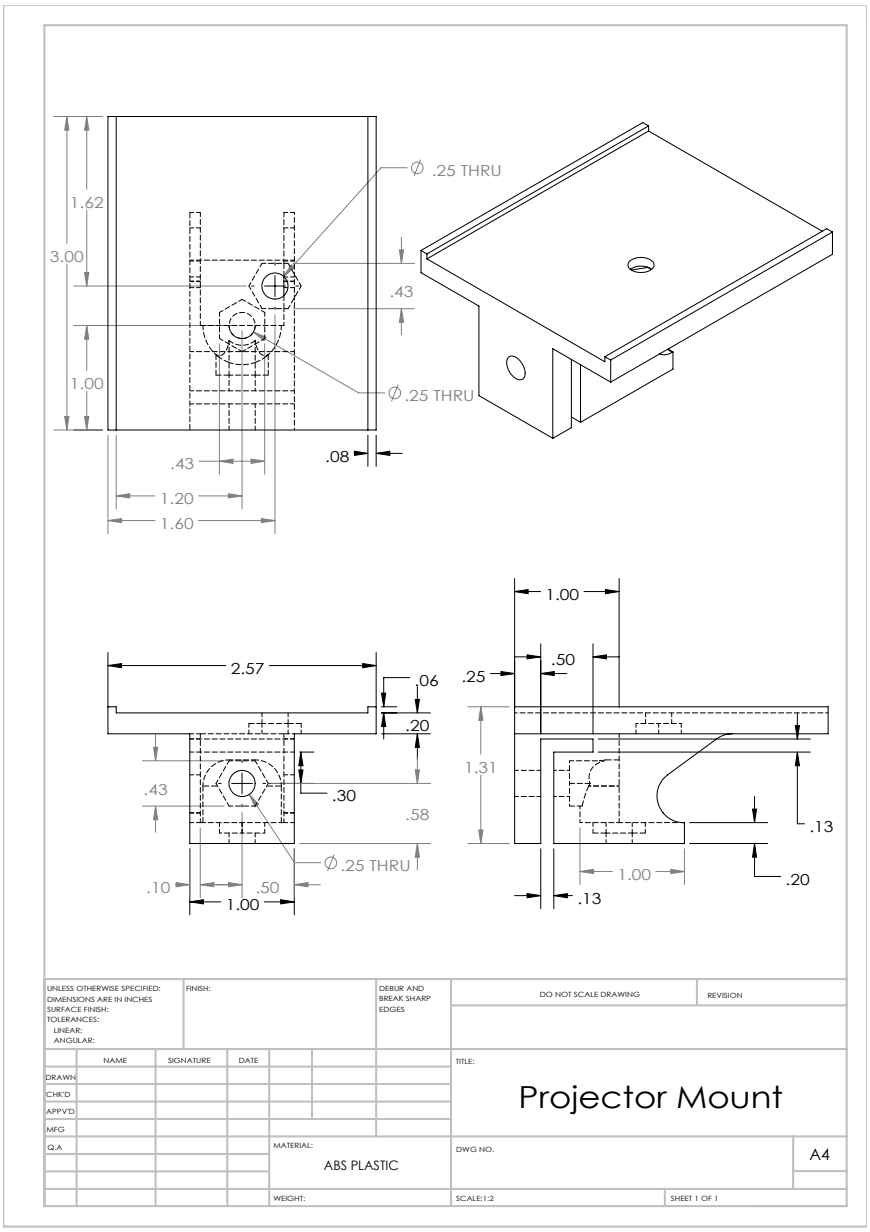


Figure E.4: Projector mount.

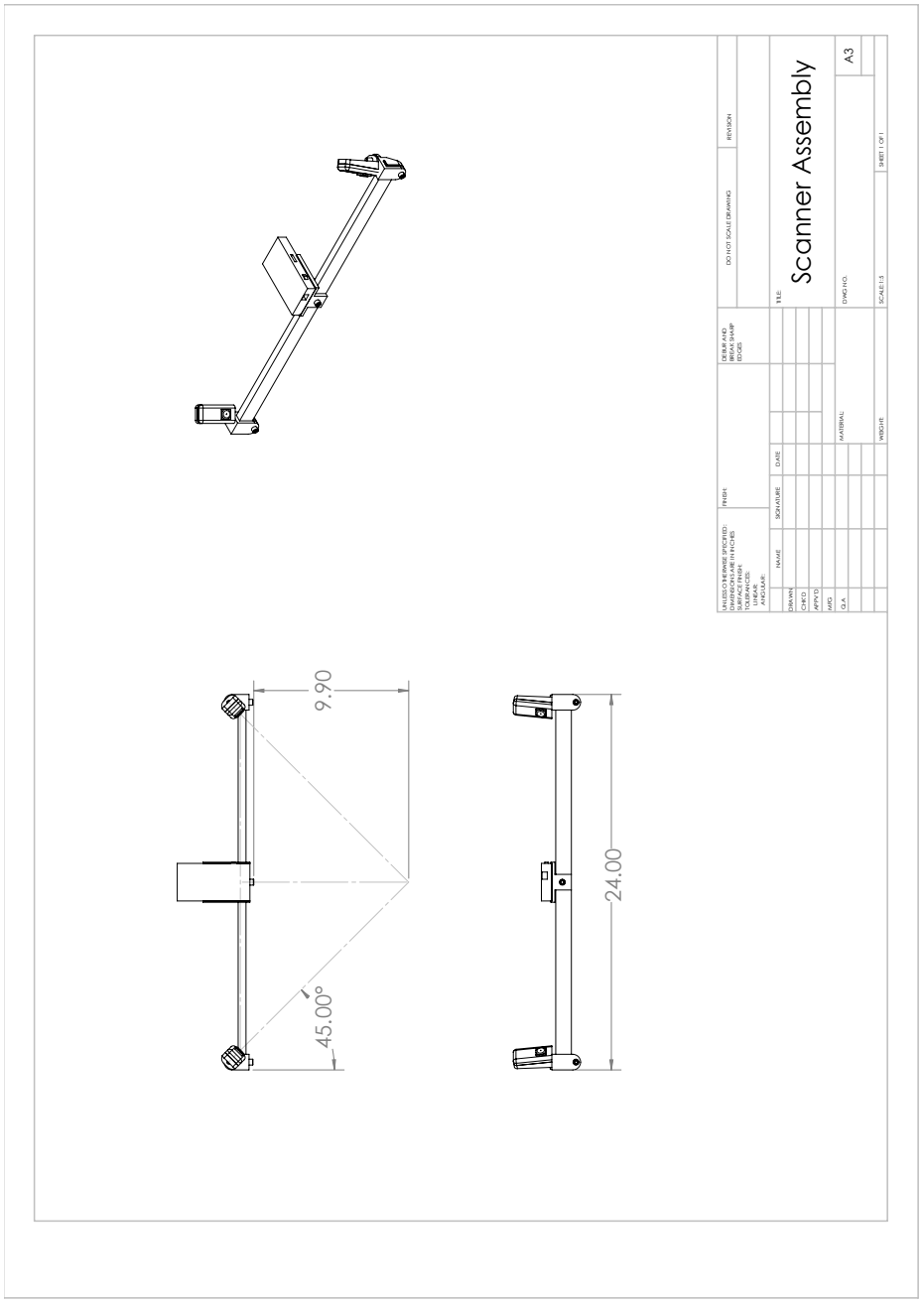


Figure E.5: Scanner assembly.

Index

References

A. Bernin. Kinect chess board meta pattern. Web page, November 2010. URL http://livingplace.informatik.haw-hamburg.de/blog/wp-content/uploads/2010/11/kinect1.png.

F. Chen, G. M. Brown, and M. Song. Overview of three-dimensional shape measurement using optical methods. *Optical Engineering*, 39(1):10–22, 2000. doi: 10.1117/1.602438. URL +http://dx.doi.org/10.1117/1.602438.

Creaform. Handy scan 3d. Online, 2012. URL www.creaform3d.com/en/metrology-solutions/portable-3d-scanner-handyscan-3d.

David LaserScanner. David LaserScanner. Online, 2012. URL www.david-laserscanner.com.

Edmund Optics. Laser scanner. website, 2012. URL http://www.edmundoptics.com/images/articles/fig-2-cti.gif.

G. Falcao, N. Hurtos, and J. Massich. Plane-based calibration of a projector-camera system. *VIBOT Master*, 9, 2008. URL http://clairethesis.googlecode.com/svn-history/r39/trunk/relatedWork/ProCam_Calib_v2.pdf.

O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. the MIT Press, 1993.

B. Freedman, A. Shpunt, M. Machline, and Y. Arieli. Depth mapping using projected patterns, 2009.

R. I. Hartley and P. Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

P. Huang, Q. Hu, F. Jin, and F. Chiang. Color-encoded digital fringe projection technique for high-speed three-dimensional surface contouring. *Optical Engineering*, 38(6):1065–1071, 1999.

itseez. OpenCV about, 2013. URL `http://opencv.org/about.html`.

J. Leno. Jay Leno's 3D printer replaces rusty old parts. *Popular Mechanics*, 2011.

Microsoft. Kinect for Windows. Web page, 2012. URL `http://www.microsoft.com/en-us/kinectforwindows/`.

NextEngine Inc. NextEngine. Online, 2012. URL `www.nextengine.com`.

OpenKinect community. Imaging information, 2013. URL `http://openkinect.org/wiki/Imaging_Information`.

Optoma. *Optoma Pico Pocket Projector PK201 Product Sheet*. Projector Central. URL `http://www.projectorcentral.com/pdf/projector_spec_5404.pdf`.

RepRap. Welcome to RepRap. Web page, 2013. URL `http://reprap.org/wiki/Main_Page`.

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

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

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

D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–195. IEEE, 2003.

K. N. Snavely. *Scene reconstruction and visualization from internet photo collections*. PhD thesis, University of Washington, 2008. URL `http://grail.cs.washington.edu/theses/SnavelyPhd.pdf`.

M. Takeda and K. Mutoh. Fourier transform profilometry for the automatic measurement of 3D object shape. *Appl. Opt.*, 22(24):3977–3982, Dec 1983. doi: 10.1364/AO.22.003977. URL `http://ao.osa.org/abstract.cfm?URI=ao-22-24-3977`.

J. R. Taylor. *An Introduction Error Analysis: The Study of Uncertainties in Physical Measurements*. University science books, 1997.

The Khronos Group. OpenGL overview, 2013. URL `http://www.opengl.org/about/`.

WeatherTech. FloorLiner™DigitalFit®. website, 2013. URL `http://www.weathertech.com/product-education-center/floorliner-digitalfit/`.

T. Weise, T. Wismer, B. Leibe, and L. Van Gool. In-hand scanning with online loop closure. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1630–1637. IEEE, 2009.

Z Corporation. 3D scanners: Selection criteria for common applications. Technical report, T. A. Grimm and Associates Inc., 2012. URL `http://www.zcorp.com/documents/111_Scanner%20Criteria%20White%20Paper%20FINAL.pdf`.

M. J. Zervas. Development of a high-speed, robust system for full field-of-view 3D shape measurements. Master's thesis, Worcester Polytechnic Institute, August 2011. URL http://files.fullyreversed.com/reports/2011/Zervas.pdf.

Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000. URL http://ieeexplore.ieee.org.ezproxy.wpi.edu/stamp/stamp.jsp?tp=&arnumber=888718.