Application Report
DLPA026–May 2011

Using DLP® Development Kits for 3D Optical Metrology Systems

Raul Barreto *DLP*

ABSTRACT

Structured light is the process of projecting known patterns of pixels onto an object. When combined with a synchronized camera, a structured light pattern generator can enable highly accurate 3D optical measurements.

This application note presents the operating theory of structured light systems, as well as the key components and design considerations. Special attention is given to hardware selection (pattern generator and camera), and choice of structured light patterns. A sample implementation using the LightCommander™ development kit is detailed to provide the reader a starting point to developing their own system.

Contents

1 Introduction 1
2 Structured Light Systems 2
3 Implementation with the LightCommander™ Kit 8
4 Conclusion 16
5 References 16

List of Figures

1 Geometric Triangulation Typically Used in Stereoscopic Systems 2
2 Structured Light System Diagram 3
3 Steps for 3D Optical Measurements 4
4 Structured Light Triangulation Between a Pattern Generator and a Camera Illustrating Ray-Plane Intersection 4
5 Pinhole Model of a Camera 6
6 Binary and Gray Code Patterns 7
7 Phase-Shifted Patterns 8
8 Structured Light System Implementation Using the Light Commander™ Development Kit 9
9 Results of Pattern Decoding 12
10 Colormaps of Measured Objects 13
11 Sample GUI to Operate the Uncalibrated System 14
12 3D Model: Metallic Screw 15
13 3D Model: Mannequin Face 15
14 3D Model and Point Cloud Visualization: Hand 16

1 Introduction

Structured light is one of the most commonly used techniques to obtain three dimensional (3D) optical measurements. The goal of a 3D optical measurement system is to capture the shape of an object in order to construct a 3D model that can be used for measurements and analysis. Applications range from machine vision systems that perform volumetric inspection in an assembly line, to biometric systems that perform 3D facial recognition.

Optical measurement systems offer various advantages over the traditional contact method where a probe contacts an object multiple times because the measurements can be taken at a faster rate and are not likely to disrupt the object. Three different techniques stand out when considering optical systems: stereoscopic, single-line scanner, and structured light. All three are generally based on the same principle of triangulation illustrated in Figure 1, but perform differently in terms of speed, accuracy and resolution.

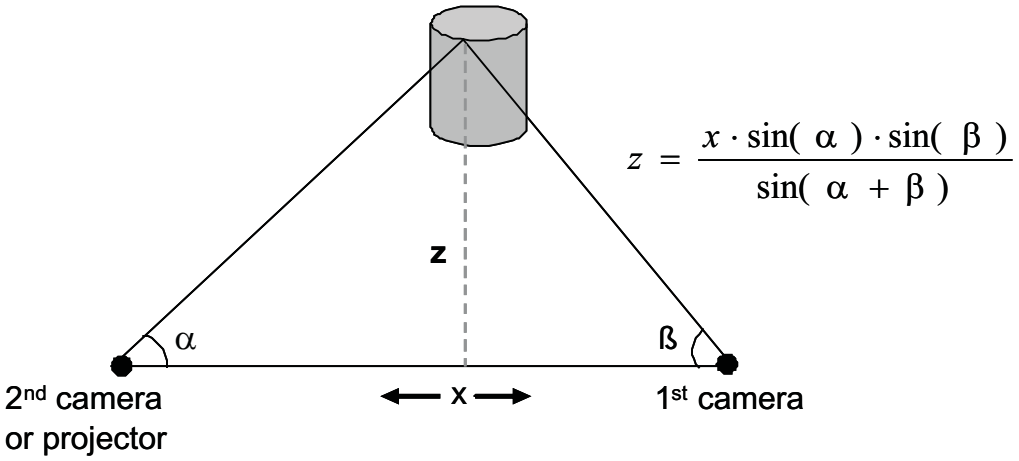


Figure 1. Geometric Triangulation Typically Used in Stereoscopic Systems

In stereoscopic systems two or more cameras image the same object from different angles and triangulation is done by finding the pixels in the different cameras that correspond to the same point. This technique can generate results quickly, but difficulties in establishing pixel correspondences make the systems less accurate. In single-line scanners, one of the cameras is replaced by a light source, usually a laser, which scans a single line of light across the object. The reflection from the object is captured by the camera and triangulation is possible because the exact angle of the laser is known for each image. The line-by-line strategy makes this type of system the most accurate, but also the slowest. Structured light systems speed up the measurement process by using a digital light pattern generator to display a known pattern of light onto the object, and a camera to capture the distortion of the pattern as it reflects from the object. The accuracy, resolution and speed depend on the type of patterns projected as well as other system considerations discussed in Section 2.2.

2 Structured Light Systems

2.1 Overview

Structured light systems consist of a digital light pattern generator and at least one synchronized camera as shown in Figure 2. A known 1D (line) or 2D (grid) pattern of light is projected and the camera is used to determine the deformation of the pattern caused by the object of interest; depth is calculated by analyzing the pattern deformation.

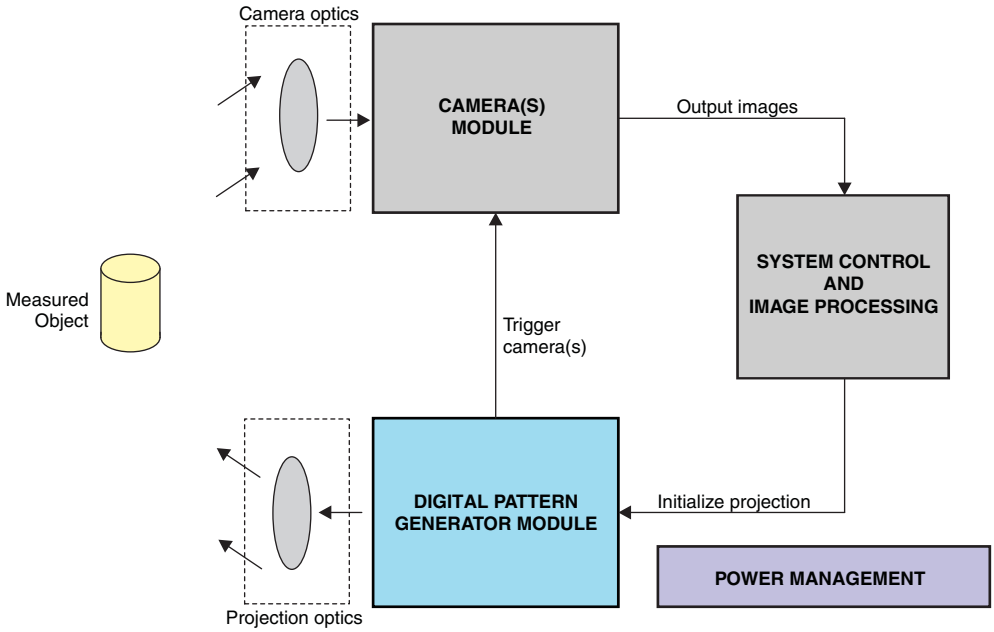


Figure 2. Structured Light System Diagram

While every structured light system is different, the sequence of steps illustrated in Figure 3 is common in most. Before projecting the light patterns it may be necessary to run a system calibration to obtain the relative position of the pattern generator and camera, improve accuracy by accounting for optical distortions, or determine the correspondence between projected and captured colors. Once necessary calibrations are established, the structured light patterns are projected and the camera is triggered to capture an image after each new pattern. Patterns are designed so that each illuminated pixel (2D), or line of pixels (1D), has a unique code that can be captured by the camera. Only areas directly illuminated by the pattern generator and imaged by the camera can be measured, so in order to measure multiple sides of an object it may be necessary to include multiple cameras on the system or rotate the object accordingly.

After displaying and capturing the patterns, the camera images are processed to obtain the desired information. The first task is to decode the patterns by matching every camera pixel or line of pixels to a unique projected pixel or line of pixels. Once those correspondences are established, the depth of each imaged point can be determined by triangulation as shown in Figure 4. The result of these calculations is typically referred to as a point-cloud, which is a dense series of unconnected points that convey depth information. For example, to measure the volume of a part or recognize a face, it is often necessary reconstruct surfaces from the point-cloud data. A common reconstruction technique is to model a surface as a polygonal mesh. There are several programs, such as GeoMagic and Meshlab, which accept point-clouds as inputs and generate reconstructed surfaces.

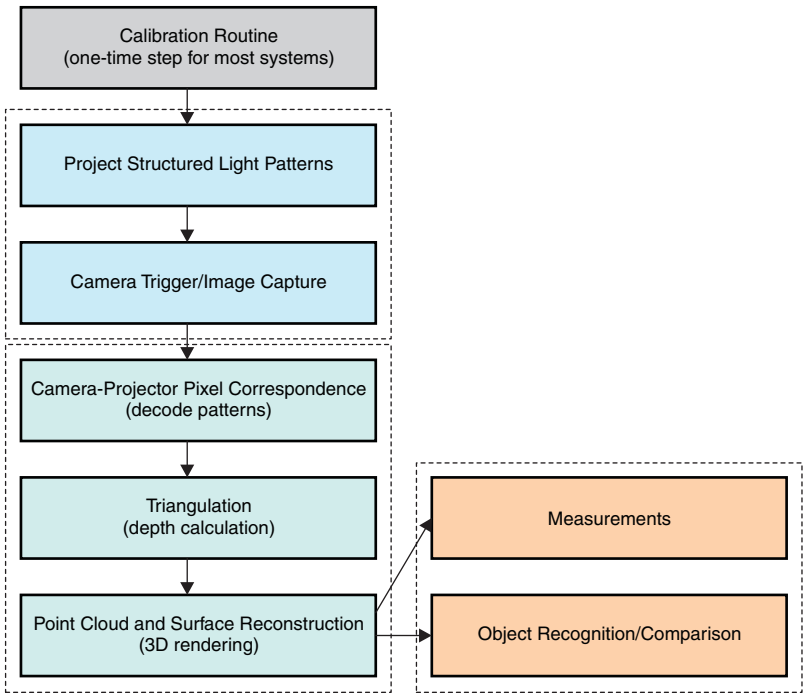


Figure 3. Steps for 3D Optical Measurements

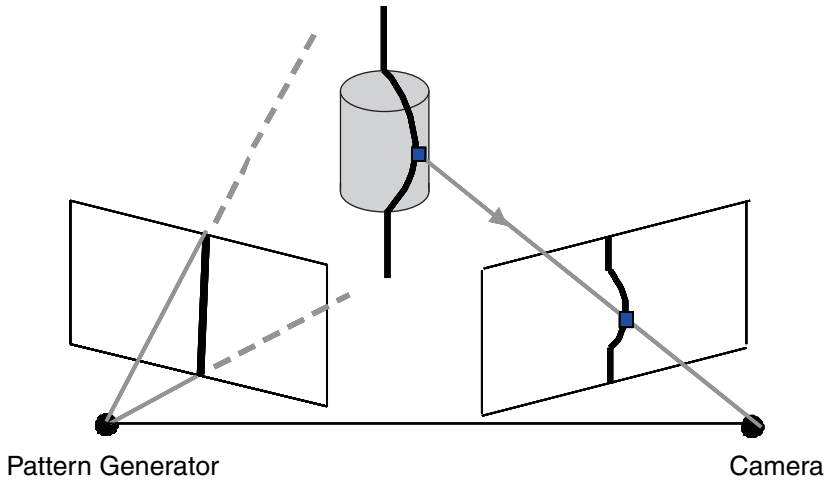


Figure 4. Structured Light Triangulation Between a Pattern Generator and a Camera Illustrating Ray-Plane Intersection

2.2 Design Considerations

When designing a structured light system it is important to consider the factors that will impact its accuracy, speed, and complexity. In this section we cover general considerations related to the selection of a pattern generator and camera, the system calibration and pattern design. There are also important considerations related to the choice of algorithms for surface reconstruction from point cloud data. Papers by Bernadini [1] and Gopi [2] provide good references for these subjects.

2.2.1 Pattern Generator

Among the most important factors to consider when choosing a digital pattern generator are resolution, speed, illumination source and optics. The required resolution depends on the size of the object to be measured (area of interest) and the minimum feature size that needs to be resolved; a higher resolution allows larger surfaces to be scanned while still resolving small features. The speed requirement, patterns per second, is driven by the maximum amount of time allowed for the system to complete a measurement and render the result. It is especially critical if a large number of patterns are required and/or the measurements will involve moving objects.

The illumination source and corresponding optical components should also be considered if particular wavelengths are required for the application. Infrared light, for example, can be used in surveillance systems when the projected patterns should be invisible to the human eye and in applications where multiple image captures must be made simultaneously without interfering with each other [3]. IR can also improve the measurement accuracy of systems where the target object reflects visible light poorly [4] or where there is high indirect illumination from the environment. These potential benefits make IR-capable pattern generators, such as the LightCommander™ kit, attractive solutions.

Other factors to consider when selecting a pattern generator include the stability and consistency of pixel values over time –in order to avoid repeated color calibrations- and the linearity of projected color gradients (for example, when the projected patterns are shifted grayscales).

2.2.2 Camera

When choosing a camera for the structured light system it is important to consider its resolution, speed, and spectral response. To take full advantage of the pattern generator, the camera's resolution should at least match that of the pattern generator. Ideally, the camera should also be able to match the pattern generator's pattern rate in order to keep up with the patterns being projected. The spectral response of the camera's image sensor, as well at its optical filters, needs to be compatible with the pattern generator's illumination source. Furthermore, when using color-coded patterns the number of hues that can be created in the specified camera capture time will affect measurement accuracy. It is also important to select a camera than can either be triggered by the pattern generator after every pattern or can trigger the pattern generator when ready to capture a new image.

2.2.3 System Calibration

Calibrating the pattern generator-camera system is important as it will directly impact the accuracy and ease-of-use of the system. In order to calculate depth through triangulation, a common coordinate system must be defined for the pattern generator, camera(s) and object of interest. External (extrinsic) parameters are used to define the relative position of the pattern generator and camera(s) or establish the correlation between pattern disturbances and object depth [5]. Internal (intrinsic) parameters can be used to account for lens distortion and improving accuracy. It is worth noting that intrinsic parameters depend only on the mechanical-optical design of the camera and thus do not change with position. Extrinsic parameters, on the other hand, need to be recalculated if the relative positions of the pattern generator and camera(s) change.

An Open Source camera calibration toolbox [6] is readily available and presents a good introduction to calibration parameters and steps using the routine proposed by Zhang [7] where the camera captures a known pattern from multiple orientations. A detailed discussion of the accuracy of various popular lens distortion models can also be found in, "A Comparative Review of Camera Calibrating Methods with Accuracy Evalution." [8]. The pattern generator can be calibrated by treating it as an inverse camera, as proposed by Lanman [9], and leveraging the same techniques used to calibrate a multiple camera system. The referenced sources make use of the pinhole model of a camera, in which the camera is described by a projection center and an image plane as shown in Figure 5. For every point on the image plane, a geometric representation of a line containing that point and the projection center can be constructed.

2.2.4 Pattern Selection

Many types of patterns have been developed through the years for structured light systems. Choice of appropriate patterns depends on spatial resolution requirements, characteristics of the object to be measured, robustness of decoding strategy, dynamic vs. static measurements, and maximum number of patterns desired.

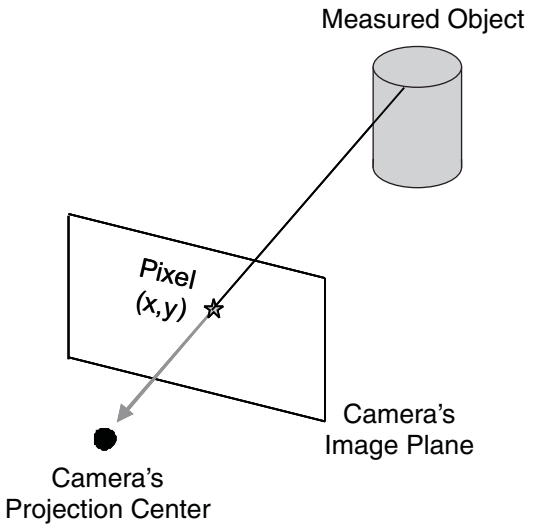


Figure 5. Pinhole Model of a Camera

Below is a summary of some of the most common pattern strategies. For a more complete discussion and quantitative comparison of pattern coding strategies, please refer to Salvi [10].

2.2.5 Binary Patterns

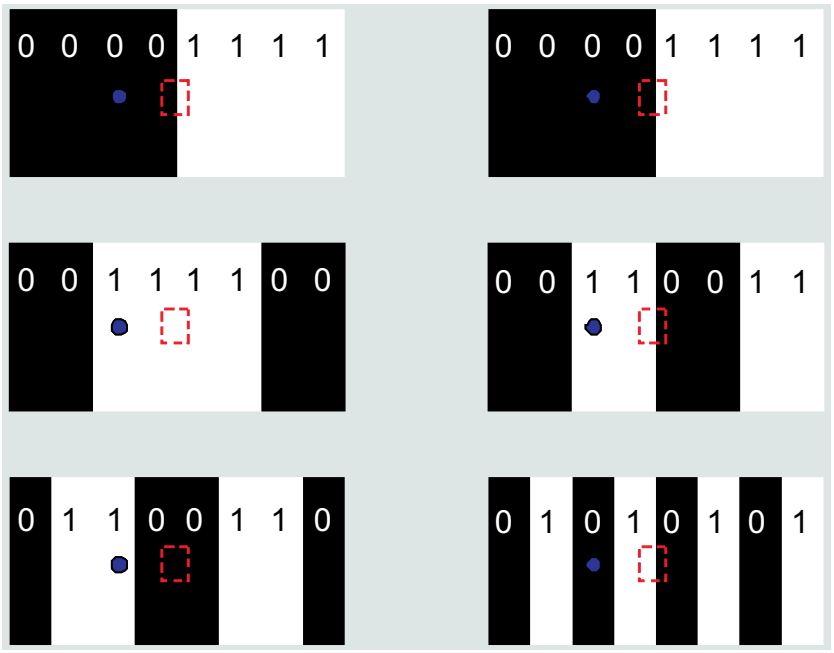
In binary pattern techniques, patterns consisting of two colors – considered 1s or 0s- are projected sequentially such that each pixel is associated to a unique digital series. Patterns are usually formed by horizontal or vertical stripes and log2(m) patterns are needed to uniquely encode m rows or columns of pixels. Because each pixel has a unique digital series, or codeword, the decoding algorithm is very robust and leads to highly accurate results. However, one limitation of this technique that should be considered is the large number of patterns required. This requirement makes it unfeasible for measurements of dynamic scenes (moving objects) and also demands a pattern generator-camera system that can operate at a high frame rate.

However, other variations of binary patterns can also be implemented to improve system performance. The most common variation is Gray Code, which makes the patterns more robust to noise and results in fewer decoding errors. When a pixel lays at a stripe boundary (change between 0 and 1), there is a possibility of misclassifying the pixel. While in binary patterns it is possible to have pixels that lay at a boundary in every pattern of the sequence; in Gray Code sequences the camera will likely see only one boundary per pixel –there is a Hamming distance(1) of one between consecutive patterns. The difference between binary and gray code patterns is illustrated in Figure 6.

Another variation is to code patterns with more than two colors or shades of gray to reduce the number of patterns. One drawback to this technique, called 'n-ary', is that it becomes more difficult for the camera to distinguish between the projected colors. This could lead to additional color calibrations in order to improve accuracy. Lastly, the technique developed by Hall-Holt [11] can be used to increase performance on dynamic objects by creating unique codes for stripe boundaries so that stripes can be followed as the object moves.

When implementing a binary pattern approach, it is important to consider how to classify pixels as 1s or 0s (illuminated or not), especially if there is significant indirect illumination in the scene. A summary of common approaches is presented by Xu [12] and includes: creating a global intensity threshold, computing adaptive thresholds for each pixel, comparing the intensity of a pattern with that of its inverse pattern (doubling the number of patterns), and constructing upper and lower bound intensity intervals.

(1) The Hamming distance represents the minimum number of digit changes (pixel values in this case) needed to change one binary string to another binary string.



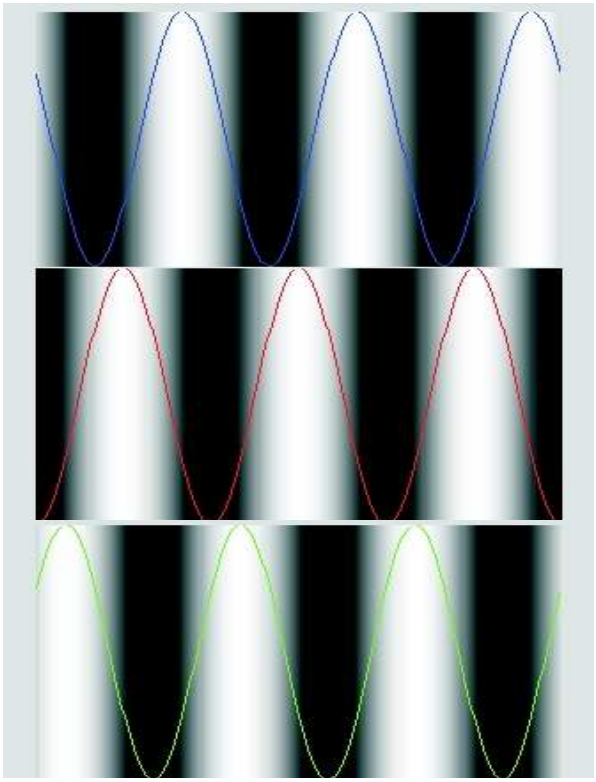
Gray	Binary
000	000
001	001
011	010
010	011
110	100
111	101
101	110
100	111

Three patterns are used to encode eight columns with Gray (left) and Binary (right) sequences. The red box illustrates the difference in pixels at stripe boundaries.

Figure 6. Binary and Gray Code Patterns

2.2.6 Phase-Shifted Patterns

In phase-shift techniques, each pattern is split into regions where the light intensity is modulated according to a sinusoidal function (e.g. modulation from black to white), Figure 7. Consecutive patterns are displayed such that the phase of the sinusoidal function shifts. A phase unwrapping algorithm then correlates each camera pixel to a pattern generator pixel. Fewer patterns are required (as low as 3) making it more feasible to measure moving objects, and resolution is actually higher than for binary patterns. The periodic nature of the patterns, however, makes correlation more difficult and can increase the number of errors. Steep slopes, such as those at the edge of an object, for example, can be challenging for phase-shifted patterns. To compensate for this drawback the number of patterns can be increased, or even combined with binary patterns, at the expense of a slower system [10]. The correlation algorithms also play a key role in phase-shifted patterns because they are more difficult to implement and can considerably affect computational time [13].



Three patterns with 2π/3 phase shift and intensity given by I = 0.5 + 0.5cos(φ +α)

Figure 7. Phase-Shifted Patterns

2.2.7 Phase-Shifted Patterns

Single-pattern techniques are best suited for measurements of moving objects, but by design, the complexity and accuracy of pixel classification make them more challenging to implement. The most common single-pattern techniques are based on De Brujin sequences, a mathematical construct to assign a neighborhood of nodes a unique color combination. Pixels are classified by inspecting their surrounding neighbors. When choosing a De Brujin sequence it is important to consider the number of color hues that can be created by the pattern generator and distinguished by the camera. In addition to De Brujin sequences, a single pattern can be designed to project a large spectrum of colors arranged by columns or rows. This technique is suitable for moving objects, but can be difficult to implement given the difficulties in correctly distinguishing within a large range of color hues.

3 Implementation with the LightCommander™ Kit

The following section describes how to implement a Structured Light system for 3D optical measurements using the DLP® LightCommander™ development kit. This implementation is intended for use in the proof-of-concept development and thus provides the user with flexibility to modify the system as needed.

3.1 System Components and Connections

The the DLP LightCommander kit is used to display the structured light patterns. This development kit provides XGA (1024x768) resolution with frame rates of 6Hz-5000Hz for binary patterns and 6Hz-700Hz for 8-bit grayscale patterns. For an introduction to the kit specs and set-up instructions, refer to http://logicpd.com/lp-lc-videos

For the second component, image acquisition, an Imaging Source industrial CCD camera model DMK

21BF04 is used. This is an industrial CCD monochrome camera, capable of capturing up to 60fps, with VGA (640x480) resolution, and FireWire interface. Although the performance of the camera does not match the LightCommander, it provides the needed capture speed for this proof-of-concept. A driver and image acquisition software are included with the camera. For detailed specifications and user manuals, refer to http://www.theimagingsource.com/en_US/products/cameras/firewire-ccd-mono/dmk21bf04/

Finally, a laptop computer with FireWire connection is used for all image processing and 3D rendering. Make sure the three components are working individually before continuing.

The first connection to make is between the LightCommander and the laptop using the USB to mini-USB cable provided with the development kit. This allows the user to access the LightCommander's Graphic User Interface (GUI) to load a set of patterns and edit the display properties. Secondly, connect the camera to the LightCommander so that each displayed pattern triggers a synchronized image acquisition. The development kit facilitates this step by providing three different output BNC ports that provide configurable synchronization signals. Use a BNC cable to connect one of these ports to the BNC input port on the camera.

Next connect the camera to the laptop in order to process the captured images. The particular connection will depend on the type of FireWire connector available in your computer. Computers with a 6-pin FireWire port are able to supply power to the camera through the same FireWire cable used to transfer data. Computers with a 4-pin FireWire port require the use of a powered FireWire hub to supply power to the camera. In this case, connect a 4-pin to 6-pin FireWire cable between the computer and the powered hub, and a 6-pin to 6-pin FireWire cable between the powered hub and the camera. When finished with all the connections, your system should look like Figure 8.

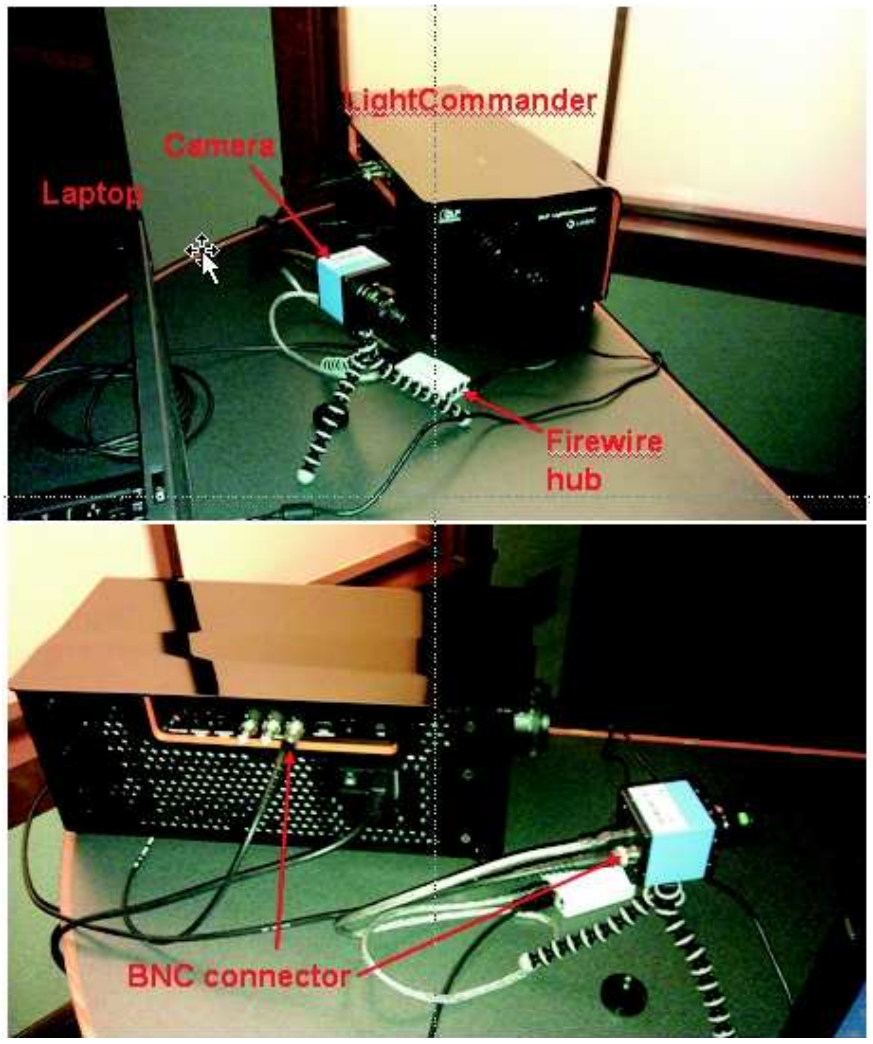


Figure 8. Structured Light System Implementation Using the Light Commander™ Development Kit

3.2 Generating and Uploading Structured Light Patterns

A sequence of Gray Code patterns, as discussed in Section 2.2 and illustrated in Figure 6, was selected for this implementation. To distinguish the 1024 horizontal pixels displayed by the LightCommander, log2(1024) = 10 vertical stripe patterns are required.

The patterns can be generated in a programming language, such as Matlab, by creating 10 binary matrixes with 1024 columns and 768 rows. Values of 0 correspond to black pixels, and values of 1 correspond to white pixels. In every matrix, all the rows corresponding to the same column have identical values (vertical stripe patterns). In the first matrix, column values switch from 0 to 1 after (1024/2^1) = 512 columns. In the second matrix, columns values switch from 0 to 1 after (1024/2^2) = 256 columns. Continue this sequence until the last matrix where values switch from 0 to 1 after every column. This set of 10 matrixes forms a binary sequence of patterns; to convert from binary to Gray Code use the following algorithm:

```
Let Bi(j) = jth column of the ith binary Pattern
Let Gi(j) = jth column of the ith Gray Code Pattern
G1(j) = B1(j)  for all j
for i=2:10
    Gi(j) = xor{ Bi-1(j), Bi(j)} for all j
End
```

In addition to the 10 Gray Code patterns, the 10 inverse patterns were also implemented to better classify the captured pixels as 0s or 1s. This strategy is discussed in more detail in Section 3.5. To generate the set of inverse patterns, invert the value of every element of the 10 Gray Code matrixes (switch from 0 to 1 and vice-versa). Lastly, an all-black (all 0s) pattern was generated and an all-white (all 1s) that used in the contrast calculation done in image processing.

Once all 22 patterns have been created, each one was saved as a bitmap image file. It is important to make sure the file dimensions are 1024x768. To upload the patterns into the LightCommander follow this set of instructions:

1. Create a new project and select Structured Light mode
2. In the Workflow tab, select Static Image Buffer data source
3. In the Timeline tab, click on Add Pattern and browse for the first saved bitmap file
4. When prompted, select 2 for Number of Output Levels and click Next
5. Load the files in this order G1, G1 inverse, …, G10 inverse, all-white, all-black

3.3 LightCommander Display Settings

With all the files loaded in the Static Image Buffer, the system is ready to start projecting and capturing the patterns. The optimal projection settings will depend, in part, on the frame rate capability of the camera, the type of lens used in the LightCommander, the camera and the background illumination conditions. In this implementation the 50mm f-mount lens that comes standard with the development kit is used. The aperture is set to "4". This aperture level provides enough brightness while also allowing objects with considerable depth to remain in focus.

To trigger the camera, enable at least one of the LightCommander Output Syncs. All three output a 3.3V signal; the polarity, delay, and width of the signal can be adjusted to match the requirements of the camera. A 20µs signal, with no delay and positive polarity to trigger the Imaging Source camera is used in the setup. The frame rate on the LightCommander set to 30fps. While the camera can support a maximum of 60fps, we found occasional problems with the data transfer resulted when the rate exceeded 36 fps.

3.4 Camera Setup

Most cameras, including the unit used in this report, have at least two operating modes: Free Running (live) and Triggered. First, verify that the camera is working properly on Free Running mode and make all necessary adjustments to Gain, Brightness, Exposure, etc. Then, with the LightCommander displaying the patterns and at least one sync enabled, switch to Triggered mode. For an initial system test, use the camera's own image capturing software to save a sequence of 22 images. It is also possible, as described in the next section, to control the image capturing using the same software that performs the image processing.

After saving a set of 22 images from the camera, the next step is to start processing the data and to derive the 3D measurements. Before proceeding to this next step, verify that the 22 image files correspond to the 22 projected patterns. If there are missing patterns or erroneous data files, consider reducing the frame rate.

3.5 Image Processing

All the image capturing and image processing for this application were developed in Matlab. A GUI was also developed at the end to provide easy access to the 3D measurement process.

The easiest way to control the camera from within Matlab is to use the Image Acquisition Toolbox, which provides a large repertoire of built-in functions to capture images. The toolbox is compatible with most common camera interfaces, but is not included with the standard Matlab package.

In this application, we used a toolbox developed by Frank Wornle of the University of Adelaide and distributed as free software under the GNU Public License Agreement. The toolbox, TIS Vision Tools, consists of a set of drivers optimized for use with The Imagining Source FireWire cameras, and provides the necessary functions to capture image frames from within Matlab. This eliminates the need to save the camera frames as picture files and import them into Matlab for processing. For detailed instructions on how to install the toolbox, refer to
http://data.mecheng.adelaide.edu.au/robotics_novell/WWW_Devs/TISVisionTools/

3.5.1 Simple Implementation – No System Calibration

The implementation described in this report does not require a pattern generator-camera calibration routine. This makes the implementation faster and easier, but is only recommended for proof-of-concept or to become familiar with the system components and image processing. This type of implementation is not as accurate and cannot achieve the same resolution of a calibrated system. It works well for determining relative depth/height differences, but not absolute values.

3.5.1.1 Capturing and Re-ordering Camera Frames

To capture camera frames, call your toolbox capture function (capImage in TIS Vision Tools, 22 times and save each frame as a matrix. If the frames are captured in RGB format, use the rgb2gray function to convert them to grayscale.

At the writing of this report the patterns run continuously from the LightCommander. There is not a “run pattern once” routine and no trigger to start displaying the first pattern. Therefore, it is necessary to re-arrange the captured frames in the correct order (the first frame captured might correspond to the 5th pattern in the sequence, for example). One way of accomplishing this is to locate the all-black frame, which is the last pattern in the sequence and should have the lowest sum of pixel values, and re-arrange the other frames accordingly. The following Matlab code implements this strategy.

```
for j=1:22            %Sum total pixel values on each frame
    pixelcount(j)=sum(sum(frames{j}));
end
[mn,ind]=min(pixelcount);     %Locate frame with lowest sum
for j=ind+1:22                %Re-arrange frames
    framesordered{j-ind}=frames{j};
end
for j=ind:-1:1
    framesordered{j+22-ind}=frames{j};
end
allblack=framesordered{22};         %All-black frame
allwhite=framesordered{21};         %All-white frame
```

3.5.1.2 Decoding Captured Frames

With the 22 patterns captured and arranged in the right order, it is then possible to determine which projected column (1024 columns projected by the LightCommander) corresponds to each pixel in the camera’s field of view

The first step in this process is to determine the binary value (black or white) of every pixel on each of the first 20 patterns. This is done by comparing every pixel on each pattern with the same pixel on the inverse pattern. If the pixel value on the inverse pattern is lower, then the pixel is classified as white (value of 1) on the original pattern. If the pixel value on the inverse pattern is higher, then the pixel is classified as black (value of 0) on the original pattern.

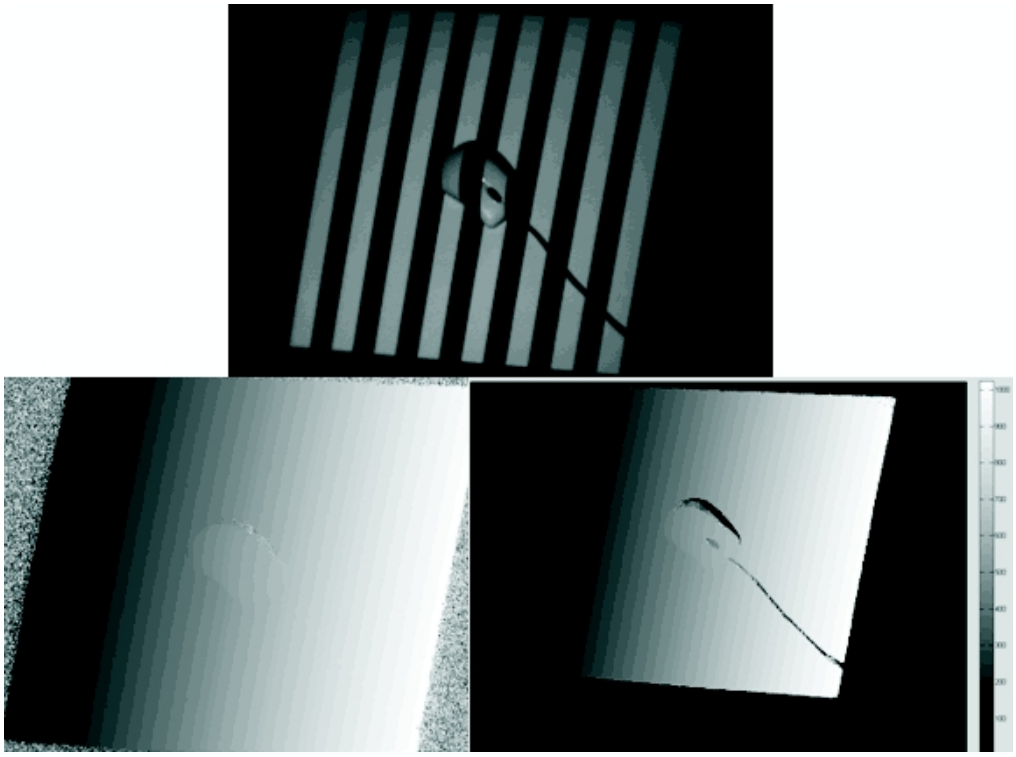
```
for k=1:10
G{k}(framesordered{2*k)-1}(:,:)>=framesordered{2*k}(:,:))=1
end
```

The next step is to convert the decoded Gray Code matrixes back to binary matrixes and from binary to decimal (values ranging from 1 to 1024). The following code performs these two conversions.

```
B{1}=G{1};
for k=2:10
    B{k}=xor(B{k-1},G{k});
end

for k=10:-1:1
    decimal(:,:)=decimal(:,:)+(2^(10-k)*double(B{k}(:,:)));
end
decimal=decimal+1;
```

The decimal values contain the correspondence between the pixels captured by the camera and those projected by the LightCommander. However, there may be regions of the camera's field of view that were not directly illuminated. In this case, we don't want the program to assign those pixels values based on the background (indirect) illumination of the room. To remove those pixels, we can calculate each pixel's contrast by subtracting the values in the all-white frame from those in the all-black frame. We then specify a minimum contrast threshold that corresponds to directly illuminated regions and eliminate all the pixels that do not meet the minimum threshold. Figure 9 illustrates the results of this decoding technique.



Top: one of the frames captured by the camera. Bottom Left: Each camera pixel is assigned a pattern generator column, encoded in grayscale, that ranges between 1 and 1024. Bottom Right: Same decoding technique with a different pixel contrast threshold to correct noisy and dark regions.

Figure 9. Results of Pattern Decoding

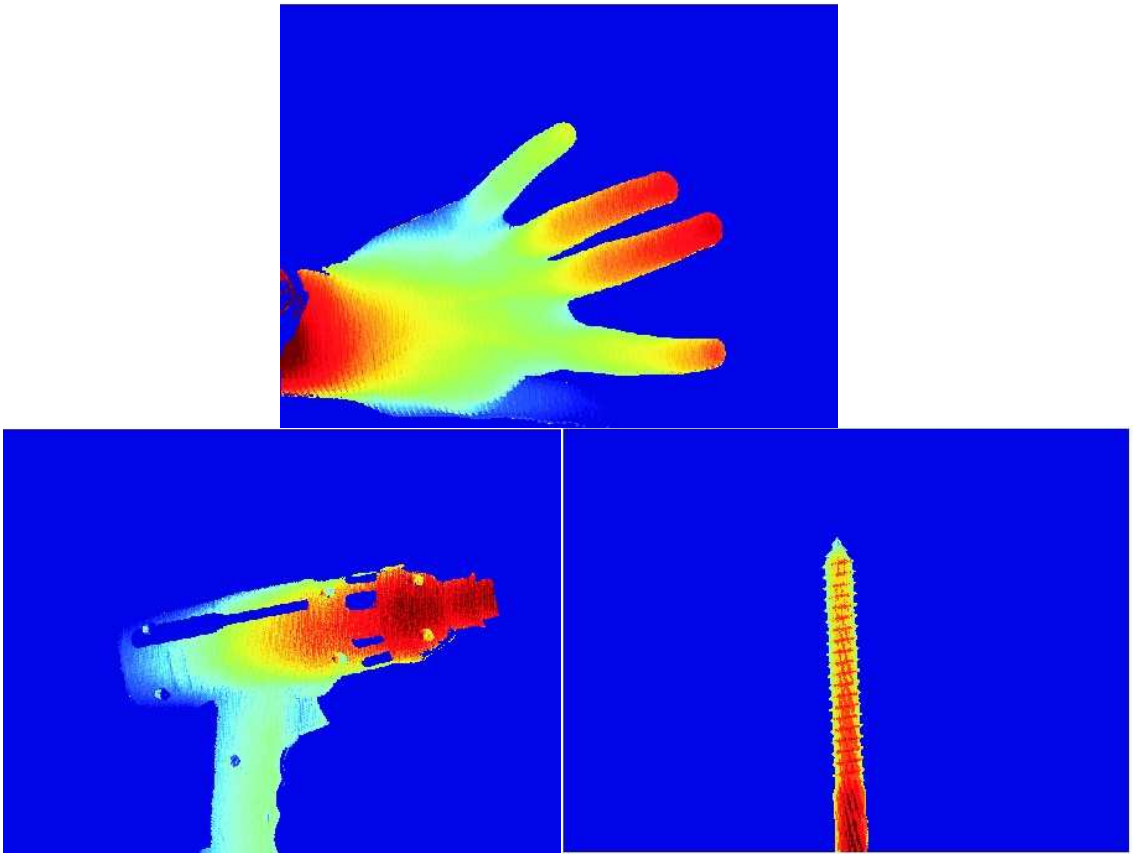
3.5.1.3 Defining a Reference Plane and Calculating 3D Measurements

The key to the implementation that does not require calibration is to define a flat surface as a reference plane and then analyze the distortion of the projected patterns caused by the object been measured. We used a foam core screen as reference object and subtracted the decoded values of the measured object from those of the reference. By imaging a series of objects with previously determined dimensions and analyzing the pattern differences between those objects and the reference screen, we can construct a scaling between pattern difference and physical measurement units. This technique generates height measurements relative to the defined reference plane.

3.5.1.4 Displaying and Exporting the Data

The output of the calculations described above is a point cloud where every data point consists of x,y,z values. One way to visualize and verify the data before constructing a 3D model (surface reconstruction) is to plot a 2D surface plot with the z-axis color coded. Figure 10 contains 2D surface plots for three different objects measured with the described technique.

Two different surface reconstruction programs, Geomagic and Meshlab, were used to create the 3D models. The data from Matlab was saved in ASCII files- each file consists of three columns corresponding to the x, y, and z values, and a maximum of 786432 rows- and imported into the surface reconstruction programs. Figure 11 shows the GUI developed in Matlab to control the camera, visualize the data in 2D, change the pixel contrast threshold, and export the data in ASCII format.



Top: left hand. Bottom Left: electric drill. Bottom Right: metal screw.

Figure 10. Colormaps of Measured Objects

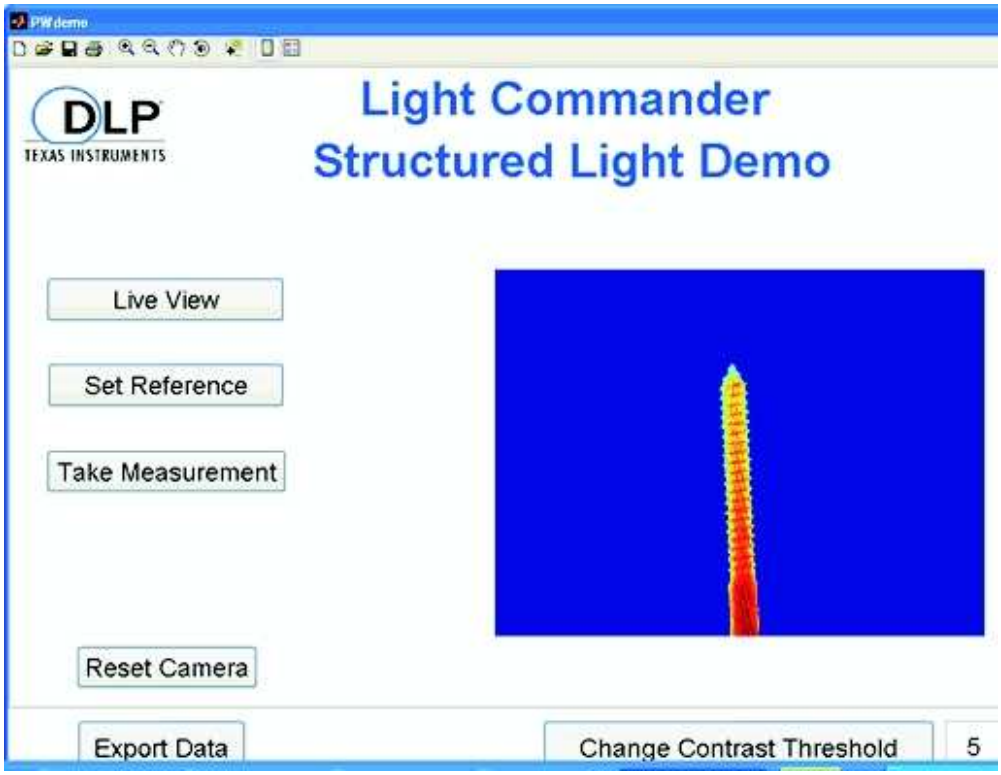
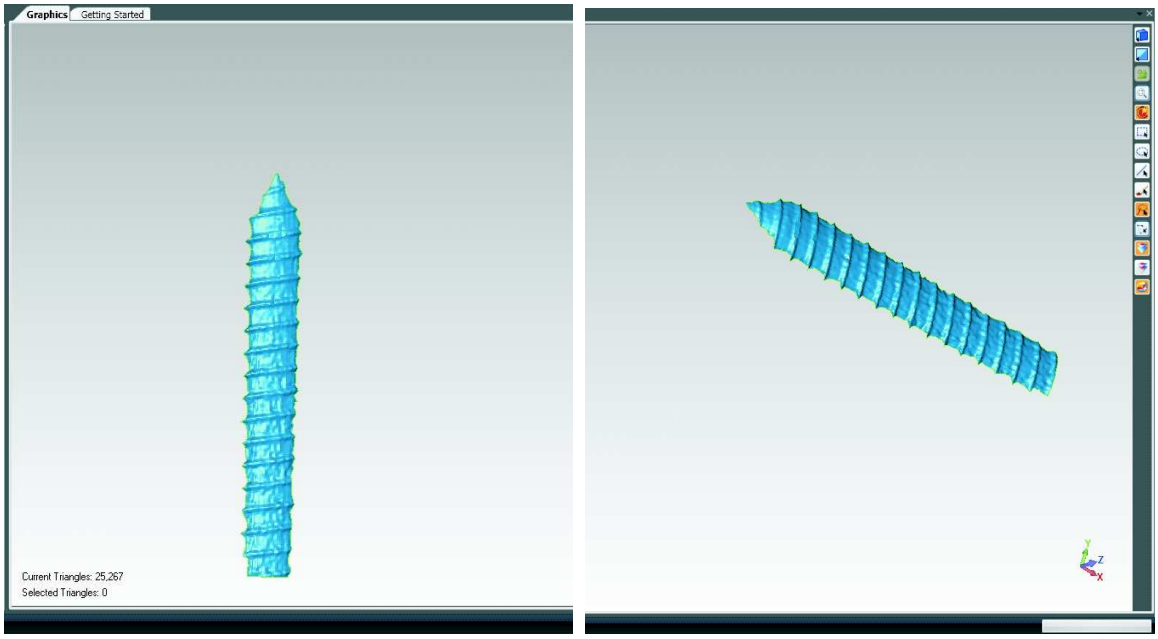


Figure 11. Sample GUI to Operate the Uncalibrated System

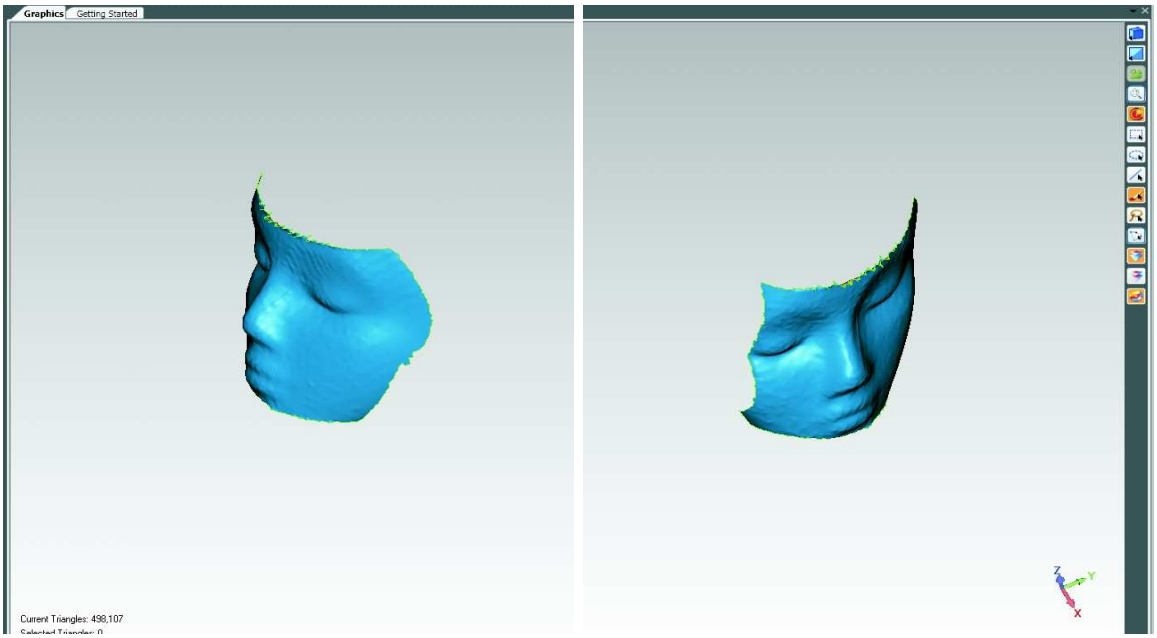
3.6 3D Rendering

This section presents the 3D models constructed for various objects using Geomagic and Meshlab software. Geomagic is a commercially available product used is manufacturing, design, and analysis [14]. The Wrap program transforms point cloud data into 3D polygon meshes in a highly automated fashion. A separate program, Qualify, performs 3D graphical comparison of objects. Meshlab, on the other hand, is a open-source program developed with support from the 3D-Coform consortium [15]. For the 3D models created in Meshlab, the Ball-Pivoting algorithm [1] was used.



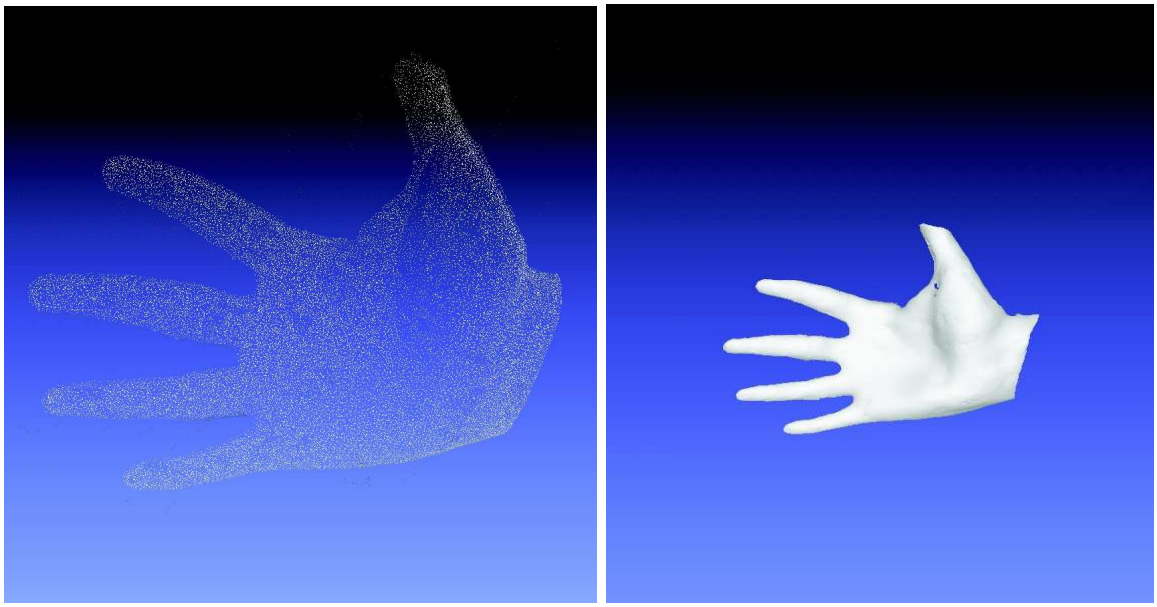
3D rendering of metallic screw created with Geomagic's Wrap software

Figure 12. 3D Model: Metallic Screw



3D rendering of mannequin face created with Geomagic's Wrap software

Figure 13. 3D Model: Mannequin Face



Point cloud visualization and 3D rendering of a hand created with Meshlab

Figure 14. 3D Model and Point Cloud Visualization: Hand

4 Conclusion

The sample implementation described in Section 3 was intended to provide a method for shortening the development time of a DLP-based 3D optical measurement system. The overview and design considerations presented in Section 2 contain helpful information to aid the developer in customizing the 3D measurement system for their unique requirements and specifications.

5 References

1. F. Bernadini, et al. “The Ball-Pivoting Algorithm for Surface Reconstruction,” *IEEE Transactions on Visualization and Computer Graphics*, Vol. 5, No. 4, pp. 349-359, 1999.
2. M. Gopi, S. Krishnan, and C. Silva. “Surface Reconstruction based on Lower Dimensional Localized Delaunay Triangulation,” *Computer Graphics Forum*, Vol. 19, pp. 467–478, 2000.
3. K. Akasaka, R. Sagawa, and Y. Yagi. “A Sensor for Simultaneously Capturing Texture and Shape by Projecting Structured IR Light,” *IEEE Sixth International Conference on 3-D Imaging and Modeling*, 2007
4. V. Paquit, et al. “Near-Infrared Imaging and Structured Light Ranging for Automatic Catheter Insertion,” *Medical Imaging 2006: Visualization, Image-Guided Procedures, and Display*, Vol. 6141, March 2006.
5. C. Brenner, et al. “Photogrammetric Calibration and Accuracy Evaluation of a Cross-Pattern Stripe Projector,” *Videometrics VI*, pp. 164-172, 1999.
6. Camera Calibration Toolbox for Matlab: http://www.vision.caltech.edu/bouguetj/calib_doc/
7. Z. Zhang. “Flexible Camera Calibration by Viewing a Plane from Unknown Orientations,” *Seventh International Conference on Computer Vision*, Vol. 1, 1999
8. X. Armangué, J. Salvi, and J. Batlle. “A Comparative Review of Camera Calibrating Methods with Accuracy Evaluation,” *Pattern Recognition*, Vol. 35, No. 7, pp. 1617-1635, July 2002.
9. D. Lanman and G. Taubin. Build Your Own 3D Scanner: http://mesh.brown.edu/byo3d/
10. J. Salvi, J. Pagès, and J. Batlle. “Pattern Codification Strategies in Structured Light Systems,” *Pattern Recognition*, Vol. 37, No. 4, pp. 827-849, April 2004.
11. O. Hall-Holt and s. Rusinkiewicz. “Stripe Boundary Codes for Real-Time Structured-Light Range Scanning of Moving Objects,” *Eighth International Conference on Computer Vision*, Vol. 1, 2001.
12. Y. Xu and D Aliaga. “Robust Pixel Classification for 3D Modeling with Structured Light,” *Proceedings of Graphics Interface*, 2007.

13. P. Huang and S Zhang. “Fast Three-Step Phase-Shifting Algorithm,” *Applied Optics*, Vol. 45, No. 21, pp. 5086-5091, July 2006.
14. Geomagic Software for Creating 3D Models: http://www.geomagic.com/en/
15. Open Source 3D Mesh Processing Software: http://meshlab.sourceforge.net/

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Audio	www.ti.com/audio	Communications and Telecom	www.ti.com/communications
Amplifiers	amplifier.ti.com	Computers and Peripherals	www.ti.com/computers
Data Converters	dataconverter.ti.com	Consumer Electronics	www.ti.com/consumer-apps
DLP® Products	www.dlp.com	Energy and Lighting	www.ti.com/energy
DSP	dsp.ti.com	Industrial	www.ti.com/industrial
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Security	www.ti.com/security
Logic	logic.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Power Mgmt	power.ti.com	Transportation and Automotive	www.ti.com/automotive
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com	Wireless	www.ti.com/wireless-apps
RF/IF and ZigBee® Solutions	www.ti.com/lprf		
	TI E2E Community Home Page		e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265