3D Surface Reconstruction for Deformation Analysis through Markerless Image Processing

Shane Transue - Digital Image Processing
University of Colorado Denver
Departments of Electrical Engineering and Computer Science

Abstract

Surface reconstruction from images based on the premise that geometric structures can be extracted using Shape From Shading (SFS) techniques is a well studied area within digital image processing. Most existing shape from shading methods focus on improving the quality of the surface that can be extracted based on improving surface curvature and depth estimations, however a new application of these techniques that has not been extensively studied is the application to surface deformation reconstructions of volumetric objects. The primary objective of this project is to: (1) explore current shape from shading techniques for surface reconstruction, (2) apply these techniques to deformation recording, and (3) improve surface curvature estimates of depth-images. Within the implementation of this project, several different techniques within digital image processing were used to reconstruct the animation of a deforming surface recorded using a Microsoft Kinect2. These include: intensity image acquisition, finite-difference based spatial convolution for image derivatives, normal map generation, and displacement map generation through field integration. The generated surfaces illustrate that the surface of a recorded deforming object can be accurately reconstructed using the proposed implementation.

Keywords: Shape from Shading, Image-based Surface Extraction, Surface Deformation Recording

1 Introduction

The process of extracting surface curvature and three-dimensional surfaces from image sequences is a field within computer vision, robotics, and image processing that has received extensive research. Many established techniques for three-dimensional surface reconstruction are currently used in camera-based vision, augmented reality, object interaction, robotic vision, and character recreation in animation and graphics. Based on these existing Shape From Shading (SFS) methodologies, this project aims to reconstruct accurate three-dimensional surfaces of deformable objects using an individual visible-light imaging device. Additionally, this technique will be extended to improve the curvature estimates of depth-images that are obtained using the same imaging device based on the lower noise corruption introduced within color-images than laser-based depth images.

The primary objective of this research is to explore how these techniques can be employed to accurately reconstruct the surface of a deforming object from a sequence of digital images that describe surface deformation behavior over time. Digital image processing has been utilized extensively for two primary reasons: (1) high resolution images can be obtained using visible-light imaging sensors compared to depth-imaging sensors (with much lower resolution), and (2) the noise potential of color images within a controlled environment is substantially lower than depth-imaging counterparts (for the selected imaging device).

Initially, synthetic intensity images generated using a three-dimensional ray-tracer will be used to verify the underlying SFS model for recording and reconstructing surface deformations of virtual animations. The Microsoft Kinect2 will then be used within this project to simultaneously capture the color and depth images of deforming surfaces using the implemented Kinect2 controller and recording software to provide experimental surface animations. From these captured surface image sequences, the minute surface deformations of real-world objects will be captured and reconstructed using color image sequences. This will provide the foundation for extracting three-dimensional surfaces of deforming volumetric objects over time and illustrate how depth images can be improved using digital image processing.

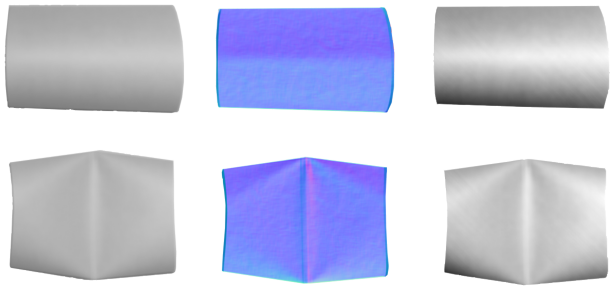

Figure 1: Illustration of the pre- and post-deformation states of a real-world surface reconstructed using digital image processing. The sequence of images illustrates the original intensity image (left), the computed normal map (center), and displacement map (right) which are integrated to reconstruct the surface of the object as it deforms. The deformation states are provided for frame [0] and frame [50] of the original animation. The post-deformation surface characteristics are clearly visible, showing this approach is effective for deformable surface reconstruction.

The project provides the implementation of an interactive three-dimensional deformation reconstruction application that provides the following functionalities: (1) synthetic intensity images can be generated using a parallel ray-tracer with support for arbitrary scene geometry, (2) real-world data can be viewed in real-time using the provided Kinect2 controller and interactive viewports, (3) real-time streams from the Kinect2 can be compressed and recorded as binary color and compressed depth image sequences, (4) the surface reconstruction process can be applied to both forms of input data (synthetic or real), and (5) the interactive replays of surface deformations with the editor.

This project has been implemented from scratch without the use of any external digital image processing libraries with the exception of OpenCV for image dilation, erosion, Gaussian blurring, and HSV segmentation (all other functions are implemented independent of OpenCV). The remaining components for intensity image processing, spatial convolution, kernels, average filtering, normal map generation, displacement image integration, and surface reconstruction have all been implemented using the provided C++ code. For other unrelated components within the application (for the interface and graphics), Qt, and OpenGL have been used. To provide the parallel implementation of the ray-tracer, OpenMP was used for CPU-based parallelization. For the real-world image acquisition the Microsoft Kinect2 has been used with the C++ SDK which provides the raw data streams from the device. Together these components form the the interactive surface deformation reconstruction application.

2 Related Work

Shape from shading and virtual object reconstruction is a field within digital image processing that has a tremendous amount of progress over the last several decades. From the original inception of the idea for extracting surface curvature based on the apparent shading of a smooth opaque object (Horn, 1970) as part of the MIT AI-lab, extensive research related to this method has been pursued within computer vision, robotics, and computer graphics.

Since the initial introduction of this methodology, several techniques (Zhang et al., 1999) have been explored for improving the accuracy (Bichsel and Pentland, 1992), quality (Han et al.,

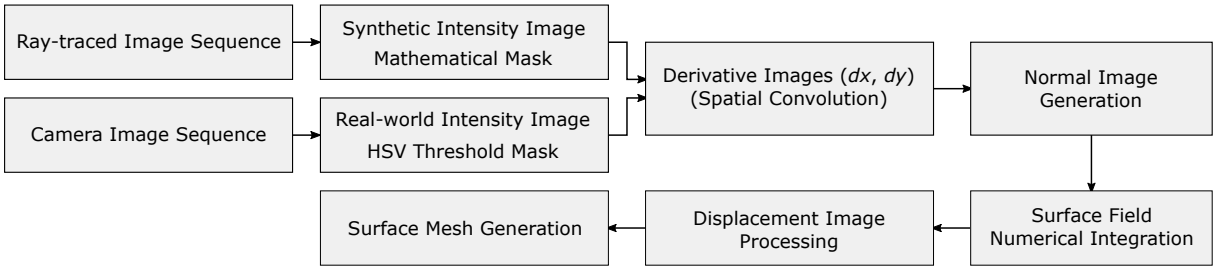

Figure 2: Overview of the presented approach and implementation pipeline. The surface deformation reconstruction pipeline is defined by five stages: (1) the source intensity image and mask sequences are provided (synthetic or real), (2) derivative images are computed, (3) normal maps are computed, (4) surface field is integrated to compute displacement map, (5) the surface of the deforming object is reconstructed into a point-cloud surface representation.

2013), and robustness (Dupuis and Oliensis, 1992) of techniques used for image-based surface reconstructions. Within the SFS formulation, image integration for surface depth reconstruction has also been thoroughly explored (Frankot and Chellapa, 1988). While none of these techniques are replicated, they all contribute to the method that has been implemented within this project one way or another (in most cases simplified).

Surface integration (Cournia, 2008) also plays a critical role in SFS surface reconstruction, however since this process is required for the process of reconstructing a virtual surface, two simple techniques have been implemented for the integration of difference fields based on the basic premise established in (Horn, 1990) with simple line integral solutions explored within (Wu and Li, 1988) and concepts from (Healey and Jain, 1984).

However unlike prior work for reducing the assumptions imposed on the general shape from shading technique (Brooks and Horn, 1985) the presented approach does not try to establish the light position or estimate scene structure, rather it is assumed that the light is in the same position as the imaging device and the surface reflectance and illumination can be loosely estimated. With this simplified model is then extended to incorporate depth-image refinement as introduced within (El et al., 2015).

3 Method

The proposed method for extracting deformable surface animations from a sequence of images is defined by six primary steps: (1) the acquisition of surface data using both synthetic and real-world images for model verification and experimentation respectively, (2) the application of standard image processing techniques for object segmentation for mask images including HSV segmentation, erosion, and dilation, (3) the estimation of the image derivatives using a Finite Differences based spatial convolution, (4) the approximation of surface normals and normal map generation, (5) the integration of the height fields extracted from the images to generate a displacement map (depth) of the shaded surface, and (6) the surface reconstruction of the recorded deformation sequence. Additionally, to extend this process to improving the quality of depth images, the resulting surface curvature data will be applied to the captured depth images to improve the visual fidelity of the depth surfaces within point-clouds.

3.1 Deformation Sequence Acquisition

The two forms of data acquisition provided are defined as (1) synthetic surface model imaging through a parallel ray-tracer and (2) the recording of video and depth sequences using a real-world imaging device (RGB + Depth Camera) The synthetic model ray-tracer has been implemented to verify the shape from shading technique with the derived formulation and to remove the noise corruption of the intensity data that is inherently included within real images. To obtain real-world images of surface deformations, the Microsoft Kinect2 was used in combination with a set of custom recording data-structures used to process the raw data from the device into the presented surface extraction method.

Synthetic Intensity Imaging Shape from shading for surface reconstruction relies on an extensive number of imposed assumptions that simplify the mathematical formulation required to make the problem well-formed for the extraction an observed surface. To further simplify the process and eliminate the potential for noise within the image, a synthetic imaging device (virtual imaging) has been implemented as a virtual ray-tracer. This synthetic form of imaging employs a simple illumination model that will be used to simulate an individual white point light used to generate an intensity image. This light source will be used to illuminate the surface of a virtual model that is then captured within this intensity image. The illumination model for this point light source follows the mathematical formulation of the *Phong* lighting model.

The Phong illumination model defines a simple three-component illumination for a point-light based on simple graphics optimizations: (1) an ambient general constant illumination value, (2) a diffuse intensity defined by Lambert's cosine law, and (3) a specular reflection component define by the relationships between the light source, surface reflectance properties, and viewing angle. Each of these components are defined below and implemented within the synthetic imaging model:

$$I_{ambient} = I_a K_a \tag{1}$$

The ambient component of the Phong model is simply defined as the product of two constants: (1) the illumination intensity of the light source I_a and (2) the reflectance coefficient of the surface K_a. The resulting value is a constant that is applied equally to all surface regions, including those occluded from the light source. The following definition of the diffuse component will then be added with this ambient term:

$$I_{diffuse} = I_d K_d (\hat{n} \cdot \hat{l}) \tag{2}$$

The diffuse component of the Phong model relates the orientation of the models surface normal vector $\hat{n}$ and the light vector $\hat{l}$ that points from the models surface at the current point to the light. Based on the cosine relationship between these two vectors, the surface intensity is then defined as the dot product between these vectors with a diffuse light coefficient I_d and material coefficient K_d that define the main illumination properties of the surface. The last component is defined by the specular component which defines a surface reflectance relationship between the light reflectance vector $\hat{r}$ and the camera viewing direction $\hat{c}$:

$$I_{specular} = I_s K_s (\hat{r} \cdot \hat{c})^{n_s} \tag{3}$$

This simply states that as the camera viewing angle and the reflection direction of the light on the surface become collinear, the light intensity is increased due to the reflectance of the surface. The exponential falloff of this intensity is defined by n_s. The final linear combination of these components defines the intensity of the ray-traced image for the pixel $I(i, j)$:

$$I(i, j) = I_{ambient} + I_{diffuse} + I_{specular} \tag{4}$$

This synthetic approach has been introduced to verify the presented shape from shading model formulation and to ensure the accuracy of the reconstructed surface with the explicit control of

the reconstruction parameters, surface reflectance and illumination model. This ray-tracing algorithm has been to implement the generation of an (n x m) intensity image where each pixel within the image casts a ray into the virtual scene to identify the exact intersection point of the surface it intersects.

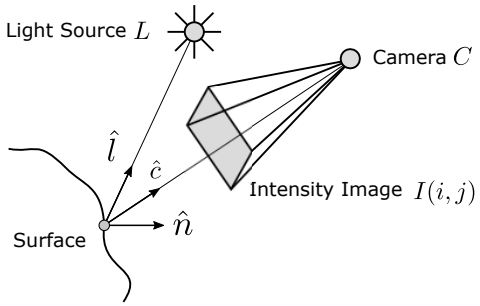

Figure 3: Illustration of the virtual scene used to generate the synthetic intensity images based on the Phong illumination model implemented through a parallel ray-tracer. The vector $\hat{l}$ represents the direction to the light source, the vector $\hat{c}$ represents the direction to the camera, and $\hat{n}$ represents the surface normal. The shaded region within the imaging plane represents the discretized region of the intensity image.

For this intersection point on the current state of the models surface, the Phong illumination model is then applied based on the position of the three-dimensional camera, its projection, the intersected surface, its associated surface normal, and the position of the light source. The resulting intensity value $I(i, j)$, defined in Equation 4, is then stored within the generated intensity image. An overview of this algorithm for an n-frame image sequence that corresponds to an n-frame surface deformation is defined below:

Algorithm 1: Synthetic Surface Intensity Ray-tracing

input : $S(t)$ - Synthetic Surface (3D Model)
L - Phong-based Point Light Source
C - 3D Camera (Imaging Device)
t - Current deformation time-step (frame t)
output: $I(t)$ - Intensity Image (ray-traced)
$M(t)$ - Segmentation Image Mask

parallel foreach *Pixel* $p_{ij} \in I(t)$ **do**
  $r_{ij} \leftarrow$ Unproject(p_{ij}, C)
  int $\leftarrow$ NearestRayIntersection(r_{ij}, $S(t)$)
  $I(i, j) \leftarrow M(i, j) \leftarrow 0$

  if *int* $\in S(t)$ **then**
    $I(i, j) \leftarrow$ *Phong*(*int*, $S(t)$, L, C)
    $M(i, j) \leftarrow 1$
  end

return $I(t)$ and $M(t)$

This algorithm provides the basis for creating artificial surface deformations using three-dimensional models as part of an n-frame animation. Thus the algorithm provided above is executed for each frame within the deforming objects surface animation (consisting of n distinct surface models). The deformation behavior observed within the synthetic imaging model can the be easily compared to the *ground-truth* model provided within the original surface animation.

Synthetic Image Segmentation The application of the synthetic illumination model for generating surface intensity images also greatly simplifies the process of image-based object segmentation. This is because within the ray-tracing algorithm, if there is no object (within the virtual scene) that is intersected, the default intensity of *zero* can be applied, simultaneously generating the object surface segmentation mask (assuming there is only one object present). This mask is then used within the synthetic surface reconstruction method where the required inputs are provide as: (1) the ray-traced intensity image and (2) the object segmentation mask generated by the ray-traced algorithm.

Experimentally Recorded Images Experimental evaluation of the shape from shading formulation presented requires several additional image processing steps. Due to the integration of noise introduced through the imaging device, lighting conditions, and complex nonlinear surface reflectance properties of real world materials, surface reconstruction from real-world images requires robust techniques for managing the as input to the surface reconstruction algorithm. These image processing techniques include: noise reduction, smooth, and reliable image segmentation. From the collection of real-world data, the applied algorithms must be sufficiently robust to handle general cases effectively, furthermore, due to the sequence of images that must be collected to describe a surface deformation, each algorithm must correctly process each image (frame) in an automated approach. Therefore all of the techniques and experimental setup defined within this section for the recorded experimental images have been integrated into an automated methodology for processing animation sequences collected using the Kinect2.

The experimental setup has been simplified to ensure some consistency with the mathematical formulation used within the synthetic model: (1) the light position and the imaging device are in the *same* position, (2) the surface deformation has been simplified and does not include occlusions or self collision, (3) the deformation object can be precisely segmented from the background, and (4) the illumination and reflectance models are assumed to be constant. The diagram below illustrates how the device, object, and light source were configured during the experimental recording of the surface deformation:

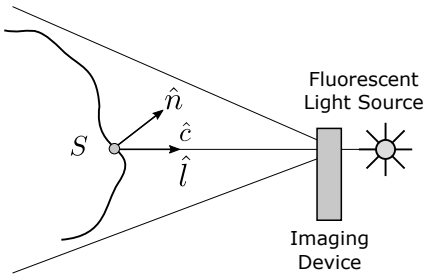

Figure 4: Overview (top view) of the experimental setup used to perform and record the surface deformation animation using the Kinect2. The light source was positioned as close as possible to the imaging device to emulate the formulation used within the synthetic model.

With the experimental setup above the imaging device was used to record two streams of data: (1) the RGB channel which composes the visible color image (1920 x 1080) and (2) the depth channel (512 x 424) which composes the surface point-cloud that defines the actual depth measurements of the surface deforming over time. Both of these data channels are recorded simultaneously and stored within custom binary animation files. The color information is saved as byte sequence encodings (RGBA) and the depth information is saved as compressed short values (based on the minimal/maximal distance of the Kinect2 which is 1.5 - 8.0[m]). The depth compression is defined as the process of converting the interval 1.5 - 8.0[m] into the short representation (2^{16}) signed equivalent. For compression: $v_c = v * 32767$, for decompression: $v_d / = 32767$. This both decreases the amount of memory and improves data storage speed for recording surface deformations. Therefore both the color and depth channels are recored and stored into two custom image sequence file types: (*.kcmov) for the RGB color movie and (*.kdmov) for the compressed depth image sequence. Within the application both of these data streams can be record simultaneously to provide the same valid data to compare between the reconstructed surface and the depth-image.

Recorded Image Masking The process of generating the image mask for the real-world data is much more involved than the synthetic form. To perform the required image processing to extract the surface of the deforming object, the object must first be identified. One of the simplest techniques for achieving this within image processing is to perform a spatial color-based thresholding. This thresholding simply defines an interval of a provided color value and then only allows pixels of that color to remain within the generated mask. Therefore the output of this thresholding process is a simple binary image that defines the pixels that belong to the surface and those that do not. However, additional image processing beyond the simple color-based thresholding is required due to the natural fluctuation of the pixel colors observed by the imaging device. Thus to ensure that an accurate segmentation of the recorded surface is obtained within the mask, this noise must be accounted for. For the recorded sequence of frames, the applied approach must also be robust to ensure that each mask will properly segment the image without large discrepancies (which would invalidate the animation). To achieve this the thresholding is used in combination with morphological image processing techniques.

The applied segmentation approach first uses the color-based thresholding technique by converting the raw RGB color information into the Hue, Saturation, Value (HSV) color space. Then a simple interval [min, max] of the accepted color range is defined within this space. Using this color interval, the accepted values are stored within a binary image such that: if the color value is within this interval then the output pixel of the binary image is 1 and if the color values is not within this interval then the current pixel of the binary image is set to 0. While this approach works well for images with low noise, there can be holes or gap regions within the generated binary image mask. Therefore to minimize these problems, *morphological* image processing techniques were used: (1) the open operation: erode(3, 3), dilate(5,5) is applied to the mask followed by the close operation: dilate(5, 5), erode(5, 5). This process fills in small holes within the mask that may filter through and ensures that the overall boundary of the segmented object is relatively consistent between each frame.

Recorded Image Smoothing An additional consideration that must be addressed for the noisy real-world data is the process of image smoothing. This is especially important due to the image derivative component of the proposed method. Since the image gradients will be computed directly from the intensity image, the gradient result will be highly influenced by the noise corrupting the image (even though experimentally this noise has a very small standard of deviation). To minimize the influence of the noise within the derivative images that are approximated using Finite Differences, two types of image smoothing techniques have been applied: (1) a Gaussian blurring filter (or low-pass filter) and (2) various sizes of simple box or average filters. These help substantially reduce the impact of the noise within the image on the resulting derivative images which contributes to more accurate surface reconstructions.

3.2 Surface Gradient

The curvature of a three-dimensional surface within the formulation of the Shape from Shading (SFS) technique is that the changes in the image intensity across adjacent pixels represents an approximation of the observed surface subject to the light and surface reflectance properties of the surface material. To begin the analysis of the surface curvature, the shading of the surface based on the provided illumination must be directly analyzed and converted into a formulated model that combines: (1) the characteristics of the lighting, (2) the reflectance properties of the surface, (3) the relationship between the surface and the light, and the extraction of higher-level surface properties using this model. The key observation of the SFS model is that while the surface geometry cannot be directly extracted from the image, if there are enough constraints about the environment, the surface, and the applied illumination, then large characteristics about the curvature and relative depth of the surface can be obtained. The first step within this process is to extract the surface curvature information based on the changes in the intensity values of the image I. These changes within the recorded intensity images can be defined as the approximation of the derivative of the image with respect to the two-dimensional domain of the image:

$$\nabla I(i,j) = \left[\frac{\partial I(i,j)}{\partial x}\right]\hat{i} + \left[\frac{\partial I(i,j)}{\partial y}\right]\hat{j} = \left[\frac{\partial f}{\partial x}\right]\hat{i} + \left[\frac{\partial f}{\partial y}\right]\hat{j} \quad (5)$$

Therefore to begin the extraction of the first order derivative of the images, the partial derivative in one direction can be approximated using Finite Differences (FD) method. Using the formulation defined above, the discrete formulation of the approximated values can be defined using one of three forms of first-order finite difference approximations: backward, forward, and central. Due to the higher order error (lower error) associated with central differences, it has been applied to approximate the required derivatives of the image in the x and y directions:

$$\frac{\partial f}{\partial x} \approx \frac{I(x+1,y) - I(x-1,y)}{2\Delta x} = \begin{bmatrix} -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \quad (6)$$

Similarly, the y derivative image is generated through the convolution of the finite differences approximation of the first order derivative. Since this process is performed within the image domain, it is also assumed that $\Delta y = 1$. Therefore the resulting equations are simplified to a set of constant coefficients.

$$\frac{\partial f}{\partial y} \approx \frac{I(x+1,y) - I(x-1,y)}{2\Delta y} = \begin{bmatrix} -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}^T \quad (7)$$

From the set of constant coefficients defined by the central differences equation are simplified into a row and column matrices, they can be adapted into image-processing kernels. Thus once these (3 x 1) finite difference coefficients are formulated, they are zero-padded into (3 x 3) convolution kernels used in the spatial convolution of the intensity image.

$$\frac{\partial f}{\partial x} \approx \begin{bmatrix} 0 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{\partial f}{\partial y} \approx \begin{bmatrix} 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \quad (8)$$

The spatial convolution of these images over the intensity images obtained from either of the image acquisition techniques introduced within the presented method will result in the approximation of the derivatives of the image for both the x and y images. The resulting derivative images (2 for every frame within the original animation) then provide discrete approximations of the surface intensity changes based on the SFS model with the simplified assumptions. Thus for each pixel within these images, the change in intensity has been obtained for two orthogonal directions based on the *observed* change in the surface height. The diagram in Figure 5 provides an illustration of these changes in the apparent height of the surface based on the changes in the intensity image. The continuous surface (dashed) is approximated by the discrete sample points (pixels) of the image.

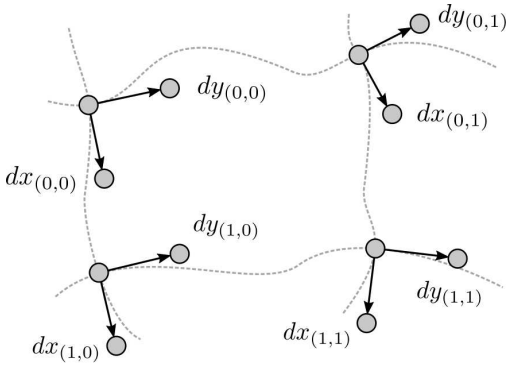

Figure 5: Continuous approximation of the surface curvature using discrete approximated derivatives. The derivatives are computed using the central Finite Differences approximation.

The vectors that describe the change in height of the surface over to the two-dimensional domain provide the first step in reconstructing the actual surface as defined by the apparent change in surface intensity from the individual light source. These differences, incorporated into the SFS model, will account for the change in height of the objects surface which will define both the surface normal approximations and the apparent depth of the surface used in the reconstruction. These two components represent the primary objectives required for reconstructing the surface of the object. The next section illustrates how the curvature of the surface is approximated using these vectors to obtain surface normal estimations.

3.3 Surface Curvature Estimation

The primary characteristic of a two-dimensional surface is its curvature. This curvature can be represented in one of two ways, each with their own purpose: (1) the change of the height of the surface or approximation of the underlying surface, and (2) the normal vectors $\hat{n}_{(i,j)}$ that define the orthogonal direction of the underlying surface. To reconstruct a virtual three-dimensional surface, the proposed method uses both of these sources as surface curvature descriptors to implicitly define a class of surfaces that adhere to the constraints imposed by the discrete samples of the first order derivative and the orthogonal surface normal vectors.

The purpose of obtaining the derivative images presented within the last section is based on the objective of providing a simple formulation of the surface normals based on the planar sampling of the continuous surface at each pixel. From this, the surface orientation at each pixel can be approximated through the simplified SFS model as the cross product between the height differentials of the derivative images. This formulation can then be simply defined for each corresponding pixel within the derivatives as shown in Equation 9.

$$\hat{n}_{(i,j)} = \left[\frac{\partial f}{\partial x}\right]\hat{i} \times \left[\frac{\partial f}{\partial y}\right]\hat{j} \tag{9}$$

Therefore the presented method defines the process of generating a *Normal Map* as using the two derivative images and the resulting cross-product to define a surface normal for each pixel. These normalized surface normal values are then converted from the unit sphere to the RGB color space such that: $r = x$, $g = y$, and $b = z$. This normal information can then be encoded into a normal image which provides an accurate approximation of the surface curvature based on this color mapping.

The formulation of this cross product also leads to the generation of a set of basis vectors for each pixel within the normal map formed by the dx, dy, and normal $\hat{n}$ vectors shown below in Figure 6. This orthogonal set of basis vectors is then stored for each pixel of each image for the entire recorded deformation.

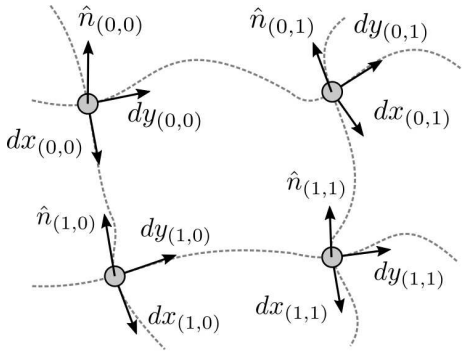

Figure 6: Orthogonal basis vectors that define the surface curvature and surface normals at the discretely sampled points derived from the proposed image-processing technique.

The collection of these orthogonal basis vectors for each discrete position across the surface of the deforming object provide temporally aligned surface deformation characteristics that can be used to reconstruct a close approximation of the surface deformation behaviors over time. While the accuracy of each of the approximated values related to the surface curvature (derivatives, normals, more sophisticated reflection model, etc.) can be improved, this formulation defines the structural requirements for replicating surface deformations based on the discretely sampled surface contained within the color image recording.

Since the surface reconstruction process requires both the surface curvature and depth information, and only the curvature information has been obtained, the next section describes the process of obtaining depth information based on the observed curvature, which is loosely defined as surface field integration.

3.4 Surface Displacement Generation

A recorded collection of intensity images defines a two dimensional domain (x, y) where each pixel has an apparent depth from the perspective of the imaging device in the direction of the surface normal. The SFS model provides a method for improving the estimation of the surface normal based on the light intensity (and direction) to provide an understanding of how light intensity values represent the underlying surface.

In the presented simplified SFS model, the assumption that the light direction $\hat{l}$ and imaging device (or camera) direction $\hat{c}$ are collinear allow this relationship to be greatly simplified such that if the surface normal is pointing directly at the light source, it should receive the highest intensity (for recorded opaque, matte surfaces), and as the intensity decreases, the surface should be approximated through the use of Lambert's cosine law. This however still only defines the surface curvature information as a function of x and y. Therefore to reconstruct a three-dimensional surface from this information using the provided simplified formulation, the relative depth of the surface implicitly defined within the orthogonal basis vectors of the surface, must be extracted.

Surface Field Integration The surface field is defined by the set of orthogonal basis vectors that are generated for each pixel extracted from the derivative images and normal map. Based on the height differentials that form this field from the derivative images, the following general form of the vector field can be defined as shown in Equation 10:

$$\vec{F}(x,y) = P(x,y)\hat{i} + Q(x,y)\hat{j} = \left[\frac{\partial I(x,y)}{\partial x}\right]\hat{i} + \left[\frac{\partial I(x,y)}{\partial y}\right]\hat{j} \tag{10}$$

From the general formulation of the vector field, there are two properties and an important outcome that can be asserted if the field is said to be *conservative*, that is the path of a line integral through the field does not modify the value of the integral:

$$\nabla I(x,y) = \left[\frac{\partial I(x,y)}{\partial x}\right]\hat{i} + \left[\frac{\partial I(x,y)}{\partial y}\right]\hat{j} \quad s.t. \quad \nabla \times \vec{F} = 0 \tag{11}$$

If the condition presented above is maintained, then it can be concluded that the closed path smooth curve integral $\oint_c = 0$ and that there exists a scalar potential function $\phi(x,y)$ corresponding to the vector field $\vec{F}$. The critical component of the formulation is how the SFS model assumptions coincide with the requirements stated by the conservative vector field with the key objective of obtaining the potential function $\phi(x,y)$ as the height potential of the surface shown as the *Integrated Surface* within Figure 12.

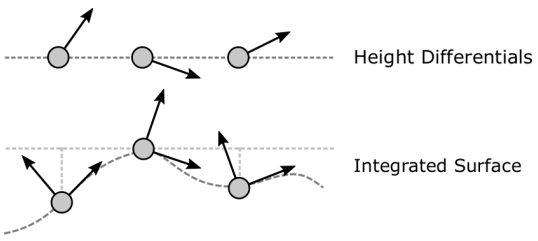

Figure 7: Illustration of the collinear height differentials (top) and the resulting integrated surface they represent (bottom). The depth from the baseline represents the scalar values of $\phi(x,y)$.

To bind the correlation between the conservative vector field and the shape from shading assumptions, the following presents the links between this mathematical model and the SFS technique.

The first assumption imposed by the SFS model is that the surface is continuous and smooth. This implies that if a point on the surface p is defined and a path P from that point across the surface back to p, the height at p should be constant (otherwise the surface will be ill-defined). Intuitively this makes sense and asserts that ideally, the closed curve integral over the surface of the object should be zero. Secondly, if the vector is conservative, then there should be a scalar potential field defined such that $\nabla\phi(x,y) = \vec{F}$. Again, this is intuitively correct because if there exists a scalar *Displacement Map* (or continuous) definition of a two-dimensional surface, the derivative exists and should be equal to $\vec{F}$ for each point (because it can be directly calculated).

While mathematically this basis can be supported, this condition cannot always be guaranteed for the applied model. Due to the noise introduced by the imaging device, the inaccuracies in the approximations of the derivatives using finite differences, and the floating-point representation, this condition may not be held. The illustration of how two independent paths below in Figure 8 through the height field can lead to two unique height values indicates that the field is not conservative.

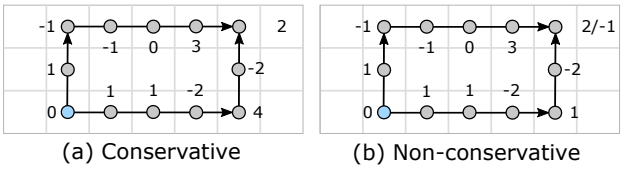

Figure 8: (a) Illustrates the potential variance of unique paths from a starting point to the end point through the field. This defines that the vector field is *non*-conservative.

However, since the underlying assertion from the SFS surface requirements state that the surface must be smooth an continuous, we continue to assume that the field is conservative, with a large error component introduced by the imaging device and the process of discretization. Within the practical sense this would indicate that $\nabla \times \vec{F} \neq 0$, however within the provided implementation, the surface integration kernel will attempt to minimize the propagation of this error throughout the computation of the displacement.

Based on the accumulation of this error, the presented model assumes that the resulting field *is* conservative, but may have a large error factor that should be minimized. Therefore the objective is to define the scalar potential function $\phi(x,y) = d(x,y)$ as a displacement scalar field based on the calculated vector field (obtained from the derivative images):

$$d(x,y) = \iint_{\Omega} \vec{F}(x,y) \tag{12}$$

This integration process however still maintains the primary problem associated with the SFS formulation: while the surface height differentials can be integrated, the constant of integration or the starting depth of the surface is unknown based solely on the height differentials provided from the gradient and intensity images. Therefore this remains a constant problem for SFS algorithms (see Section 4.1 about the depth-assisted SFS methodology).

There are other formulations of this problem that allow for computationally accurate results. Alternatively, the problem can be formed into a large optimization problem subject to the constraint that the scalar potential field $d(x,y)$ must provide height differentials that are orthogonal to the surface normals for each normal $\hat{n}_{(i,j)}$. While a system of linear constraints can be used to solve this problem for sufficiently small domains, the formulation of the system becomes extremely large as the image size increases, making this approach somewhat less feasible. Therefore, to sacrifice accuracy, simpler techniques for addressing this problem have been devised.

This section introduces some of the simplest forms of approximating the displacement function $d(x,y)$ from the provided vector field and two have been implemented to compare the results of two simple integration approximations and their impact on the resulting reconstructed surface. The most common surface traversal techniques used to compute the displacement of the surface are presented below in Figure 9:

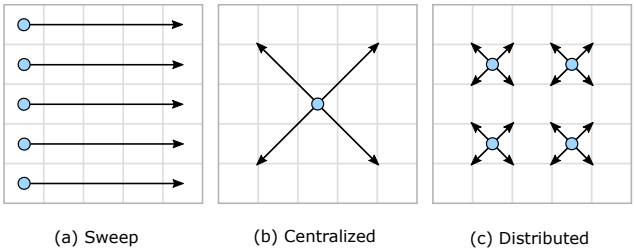

Figure 9: Illustration of the three primary methods for approximating the integration of the field based on various traversal techniques. (a) presents the sweep technique that is the simplest to implement, but has large error accumulation, (b) propagates height outward and reduces error accumulation by half, and (c) provides a distributed model used by most statistical-based techniques.

The primary problem associated with the approximation techniques shown in Figure 9 is that the error accumulation can result in substantial image artifacts which will contribute to large errors within the reconstructed surface. This is especially true for traversals that sweep the entire image. Within the proposed approach, the techniques from Figure 9 (a) and (b) have been implemented and compared[1].

Image Integration Paths Two of the most efficient integration techniques utilize simple spatial traversal schemes over the height differentials (similar to a spatial convolution). As the filter or height changes are propagated, the unknown height values will be filled in over the surface of the image. The presented method first illustrates: the sweep approach commonly used within the original SFS models and then presents a new alternative *integration* kernel based on the spiral indexing scheme with a weighted, directed graph used to represent height differentials with averaging and no-overlap propagation.

The simplest form of integration to obtain the displacement heights of the observed surface is based on a simple one-dimensional trapezoid integration scheme. The premise of this approach is simple: start at a predefined height h_0 for each row or column within the image, then using the height differentials in the x-direction, simply add the height changes and progress to the right side of the image. The height of the current column j is then the sum of the original height plus all of the height differentials to that point. The following equation formulates this change in height for any column within the image starting from the left side as shown in Figure 10 (left).

$$q_i = q_0 + \sum_{i=0}^{j} dx_i \text{ for } j = 1,2,3,...,w \text{ where } q_0 = h_0 \tag{13}$$

This approach is immediately flawed due to the error accumulation that builds over the entire image. The error component within each sample in the current row is accumulated to the final result. This error growth also drastically decreases the accuracy of the depth for the right side of the image due to this error accumulation. Similar approaches have been proposed for using this technique but sweeping across the image in four directions (right, down, up, left) with averaging, however since the method still uses large image sweeps, the error accumulation is still persists.

[1] Statistical techniques for distributed propagation models for computing the displacement map have been published but the implementation was out of the scope of this course, but provide much more reliable and accurate results.

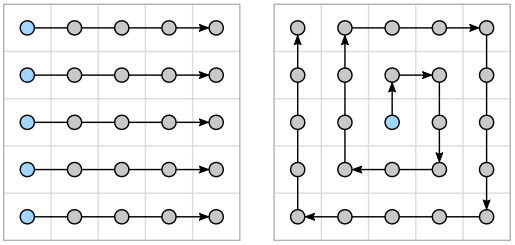

Figure 10: Illustration of the indexing schemes used to evaluate the integration of the surface height over the spatial domain of the normal image.

The critical component for any spatial integration scheme that traverses over the image is to control the propagation of the error introduced in each height differential. Additionally, if the field is not conservative and contains paths with different heights, they should be averaged in an attempt to minimize the error shown within the resulting surface (since it cannot be completely avoided).

The primary integration technique implemented within the presented surface reconstruction method is based on the centralized propagation technique shown in Figure 9 (b) and follows the spiral indexing scheme shown in Figure 10 (right). This ensures that both dx and dy derivatives are used to provide accurate height differentials and the error propagation is cut in half when compared to existing techniques that use full image sweeps. The implemented integration kernel is defined as a (3 x 3) directed weighted graph that uses the orthogonal basis vectors presented within Figure 6 as the height differentials to compute the changes in height from the center of the image (which starts out with a default height of $h_0 = 0$). The directed graph diagram with associated signs and vertex labels is provided within Figure 11.

The key characteristic of the spiral integration kernel that distinguishes itself from a normal spatial kernel is that it uses the center kernel value to determine the surrounding adjacent surface heights rather than using the neighboring values to compute the center value. This behavior is required because we assume that we know the center height to be defined as the preset integration constant h_0. From this constant value, which can be estimated through SFS albedo or accurately measured using a depth-image, the height differentials of the surrounding neighboring pixels define how the height of the surface changes with respect to this predefined value. Therefore this the execution of this kernel does not define the same convolution process as a normal filter kernel, but rather propagates the height differentials from the center of the image to the outer border of the kernel. This process provides an accurate means of propagation for the height differentials over the surface of the sub region and also reduces the overall image error accumulation by half when compared to the naïve row integration.

$$
\begin{aligned}
H_a &= \frac{1}{2}\big[(-be)+(-ab)+(-de)+(-ad)\big] \\
H_b &= (-be) \\
H_c &= \frac{1}{2}\big[(-be)+(bc)+(ef)+(-cf)\big] \\
H_d &= (-de) \\
H_e &= H \\
H_f &= (ef) \\
H_g &= \frac{1}{2}\big[(-de)+(dg)+(eh)+(-gh)\big] \\
H_h &= (eh) \\
H_k &= \frac{1}{2}\big[(ef)+(fk)+(eh)+(hk)\big]
\end{aligned}
$$

The equations above define the height differentials based on the paths through the directed, wighted graph of the kernel. Based on the traversal from the center node (with known height), the signs of the height changes are defined by going *along* or *against* the directed edges of the graph.

As the kernel from Figure 11 starts from the center of the image $(h/2, w/2)$, it traverses in a spiral towards the outer border of the image. For each index in the spiral the kernel is evaluated. This evaluation process is as follows: (1) get the current height at node e as H, (2) from node e traverse to each node (ex, $e \rightarrow a$, $e \rightarrow b$, etc.) for all 8-neighbors, (3) if moving along the edge, the height differential sign is position (+), otherwise the height differential sign is negative (-), (4) for each node that has *not* yet been assigned a value, compute the relative height and assign it.

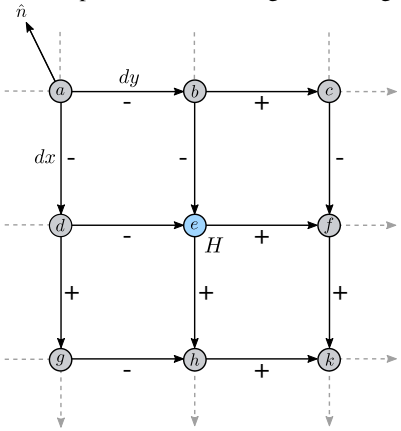

Figure 11: Weighted, directed graph that illustrates how the neighboring heights are evaluated using the 8-neighbor integration kernel. Since nodes a, c, g, and k have more than one path, their two path weights from e are computed and averaged.

Algorithm 2: Displacement Image Integration

input : $dx(t)$ - Surface Derivative Image (∂x)
$dy(t)$ - Surface Derivative Image (∂y)
t - Current Deformation time-step (frame t)
h_0 - Height Offset (integration constant)
output: $D(t)$ - Displacement Image (integrated field)

$s_{ij} \leftarrow$ SpiralIndexList $\leftarrow$ Spiral(*height*, *width*)

foreach *Valid Pixel* $s_{ij} \in D(t)$ **do**
  $ab \leftarrow D(s_{i-1j-1},t).y$
  $ad \leftarrow D(s_{ij-1},t).x$
  $bc \leftarrow D(s_{i-1j+1},t).y$
  $be \leftarrow D(s_{i-1j},t).x$
  $cf \leftarrow D(s_{i-1j+1},t).x$
  $de \leftarrow D(s_{ij-1},t).y$
  $dg \leftarrow D(s_{i+1j},t).x$
  $ef \leftarrow D(s_{ij+1},t).y$
  $eh \leftarrow D(s_{i+1j},t).x$
  $fk \leftarrow D(s_{i+1j+1},t).y$

  $H_a \leftarrow \frac{1}{2}\big[(-be)+(-ab)+(-de)+(-ad)\big]$
  $H_b \leftarrow (-be)$
  $H_c \leftarrow \frac{1}{2}\big[(-be)+(bc)+(ef)+(-cf)\big]$
  $H_d \leftarrow (-de)$
  $H_e \leftarrow D(s_{ij},t)$
  $H_f \leftarrow (ef)$
  $H_g \leftarrow \frac{1}{2}\big[(-de)+(dg)+(eh)+(-gh)\big]$
  $H_h \leftarrow (eh)$
  $H_k \leftarrow \frac{1}{2}\big[(ef)+(fk)+(eh)+(hk)\big]$

return $D(t)$ and $M(t)$

This process has been formulated within a single algorithm (shown above) that is performed using the following equations to determine the height for the neighboring pixels of the current spiral index.

This effectively propagates the height changes of the derivative images from the center of the displacement map to the edges of the image, generating the required displacement function $d(x,y)$ to reconstruct the surface of the deforming object. This process is constant based on the geometry of the kernel provided in Figure 11, therefore the new heights of the nodes $a-k$ can be statically defined. The spiral integration kernel algorithm is presented based on the diagram shown in Figure 11.

The resulting displacement map represents the approximation of the scalar potential $d(x,y)$ which can be used to approximate the depth of the surface during the reconstruction. Each value within the displacement map is only computed once for each (x, y) pair, therefore if the same index is accessed again (revisited), a new height will not be computed. Therefore the height propagation expands outwards without the introduction of additional height discrepancies.

4 Surface Reconstruction

Surface reconstruction of a three-dimensional model extracted from the SFS model is based upon three requirements: (1) the underlying structure of the geometry - in this instance it is assumed that the lattice structure of the image is maintained to generate the surface, (2) the height of each point within this lattice structure - provided by the displacement map, and (3) the surface normal defined for each point within the lattice. Thus each generated point is defined by a pixel coordinate (x, y), a height coordinate (z), and a normalized surface normal $\hat{n}$. Using these components a three-dimensional surface can be generated and illustrated as a point-cloud or triangulated surface. This surface can then be lit using traditional graphics shading to provide a clear representation of the reconstructed surface.

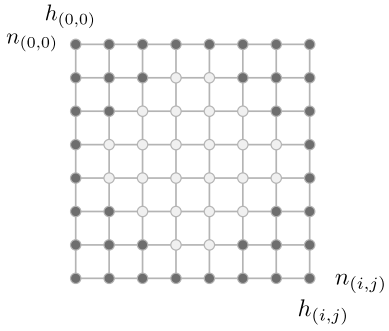

Figure 12: Illustration of the mesh lattice that corresponds to the width and height of the images within the reconstruction sequence. Each node within the lattice structure contains its own displacement value and surface normal to define the depth and curvature of the surface. Depending on the mask images (shown as darkened nodes), the total number of surface points within the surface reconstruction will naturally fluctuate.

4.1 Depth-image Assisted SFS

To improve the quality of a depth image for the recorded deformation, the implemented SFS technique can be used to more effectively estimate the surface normals for each depth sample than leading k-neighbor based approaches (such as those within the Point-Cloud Library PCL). This is because of the additional surface curvature information that is not directly available within the depth image. Since the surface normals of depth images are commonly estimated, the presented approach has been applied to improving the surface normal estimations of the depth image simultaneously captured from the Kinect2.

Improving the surface reconstruction provided by depth images can be easily achieved after the orthogonal basis vectors of Figure 6 are computed. This highly accurate description of the surface curvature for each frame within the sequence can also be applied to the depth images that were captured at the same time (the implementation records these two data streams simultaneously for this purpose). The surface normals of the point cloud are then directly mapped from the color-based analysis to provide the approximate surface normals. Since the depth image is much lower resolution than the color image, the resolution of the color image has to be mapped to the depth image, which significantly reduces the recorded resolution; however, the normals extracted using the presented SFS technique still provides a drastic improvement over the k-neighbor normal estimation technique presented within PCL.

5 Implementation

The implementation is composed of seven library modules and one primary application. Each of the supporting libraries has been developed specifically to support the functionality introduced within the presented method. Each of the supporting modules listed below are linked together to form the final form of the application.

Core Library Basic functionality for reading and writing binary files. These were written to simplify the process of loading and saving the binary representation of the Kinect2 data streams for color and depth. These operations have to be as fast as possible to ensure an adequate frame-rate for recording a deforming surface and to ensure that large animation sequences are written directly to disk.

Deformation Library This module contains the combination of all the other libraries linked together to perform the algorithms presented within this work. The primary class within this module that performs all of the core algorithms is the *DeformationAnimation*. This includes: (1) generating mask images, (2) applying averaging filters, (3) estimating image derivatives, (4) generating normal images, (5) integrating the basis field to generate the displacement map, and (6) generating the surface point-cloud for each frame in the recorded sequence.

Graphics Library The graphics library provides support for loading and rendering *.obj files, point-clouds, the bounding hierarchy for improving the performance of the ray-tracer, and several other utilities. The main graphics components within the implementation are simply used for rendering images, meshes, and point-clouds.

Image Library The image library contains all of the image-processing related work for this project. This includes classes for storing images, intensity images, kernels, normal maps, and OpenCV bindings for basic functions (HSV masking, Gaussian Blur). These were the *only* two image processing functions that were not implemented from scratch, everything else was implemented within the project.

Interface Library This module contains all of the custom Qt interface code used to create all of the custom viewports used throughout the application. Viewports have been created for each type of data that can be displayed within the interactive editor including: (1) intensity image viewports, (2) image (RGB) viewports, (3) mesh viewports, and (4) point-cloud viewports.

Kinect Library The Kinect module provides the C++ code used to access raw Kinect2 data from the Microsoft Kinect2 SDK API. This library includes the implementation of a Kinect2 sensor with the ability to monitor real-time color and depth streams and record them into large binary movie files. The implemented movie files (*.kdmov) for depth and (*.kcmov) are custom file-types aimed at improving the performance of the deformation image capturing algorithms.

Math Library Templated linear algebra library used heavily throughout the entire project for graphics, image processing, and most algorithms. This library provides an extensive number of graphics utilities that are used within the virtual scene and ray-tracer implementation.

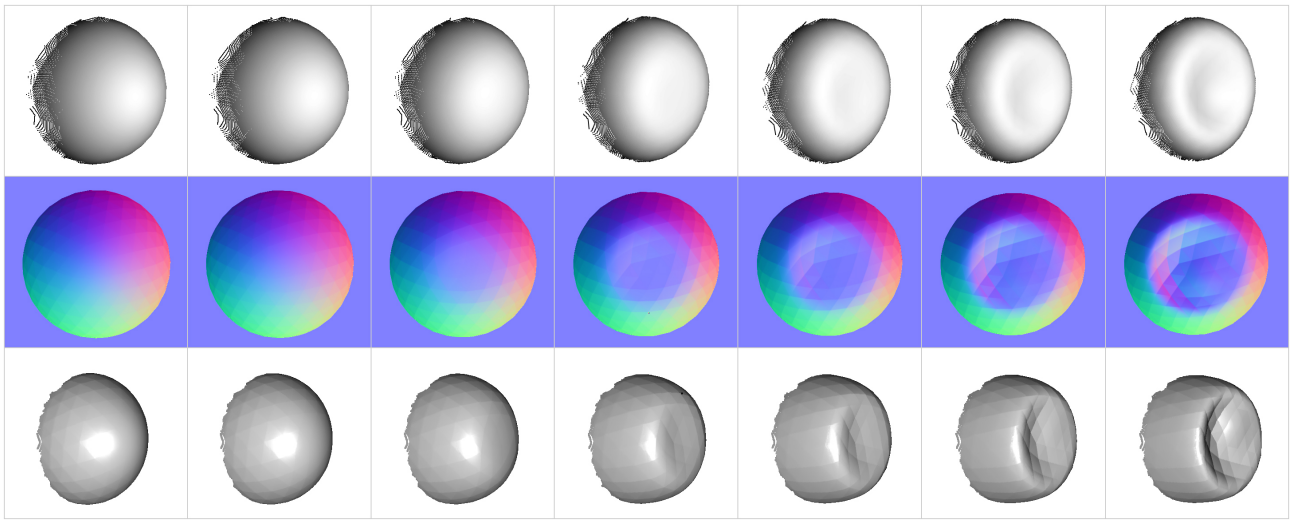

Figure 13: Overview of the image sequence that defines the synthetic deformation of a volumetric object using artificial force to cause the indentation. The (top) row illustrates the ground-truth of the deformation which has been obtained by the mathematical intersection of the surface during the ray-tracing procedure, the (middle) row illustrates the resulting normal map for the corresponding frame, and the (bottom) row illustrates the surface reconstruction of the surface. Important: the derivatives in this reconstruction were approximated using the Sobel filter.

Synthetic Deformation The main application that uses all of the modules above to implement the final project. The main application uses all of the preceding modules extensively to provide an automated interface for the surface reconstruction method presented within this work. The application has the ability to interactively generate synthetic deformations based on *.obj sequences or based on the Kinect2 stream that can be used directly within the program.

The interactive editor of the application contains four main viewports that can be used for the reconstruction of both the synthetic and real-world data. If the synthetic images are to be generated, a directory containing a sequence of *.obj meshes can be selected. Once the directory is selected, all *.obj files will be automatically loaded and formed into an animation. Once the animation is loaded, the animation can be ray-traced using the main UI controls. For the ray-tracing process, each frame will be ray-traced using the current camera position and screen resolution (the size of the image will be equal to the viewport size when the animation is rendered). For the real-world image sequence, the Kinect2 can be loaded and the depth + color stream can be recorded simultaneously. Once the real-world data is recorded and loaded, it can be used to reconstruct the surface of the object using the color reconstruction editor.

6 Results

The results of the presented Shape from Shading technique for deformable surface reconstruction are based on both the synthetic and real-world images. First the SFS model is verified using the synthetic data based on a sequence of images that are obtained using the ray-traced intensity images that contain no noise. These images are initially used because additional steps related to image blurring and segmentation are not required. The same surface reconstruction algorithms will then be applied to the real-world images collected using the Kinect2 with the addition of the HSV color segmentation, morphological image processing for improving the mask images, and the average filtering used to improve the computation of the real-world image derivatives.

6.1 Synthetic Results

The synthetic results are derived from the ideal constraints of the SFS model obtained using a ray-traced intensity image of the surface of an object deforming over time. The synthetic images are used due to the additional complexities involved with recording and reproducing surface deformations. One of the primary challenges with recording deformations is the requirement of inducing a physical force on the object to make the surface deform. Visually this is a challenge because as the surface is being recorded, the means of imposing the force on the object must not interfere with the visually collected images. Therefore the synthetic imaging implementation eliminates this problem by simply defining an animation from a set of discrete surface states that describe the deformation over time (artificially). The second challenge that is addressed with synthetic images is the elimination of the noise that naturally corrupts real-world images from imaging devices. Most shape from shading techniques are sensitive to high levels of noise which can greatly degrade the surface extraction process. Thus to simplify this process for recording deformations, a ray-tracing algorithm has been provided to generate noise-free images. The image sequence below in Figure 15 illustrates the ordering (left to right, top to bottom), of the images that are generated using this approach.

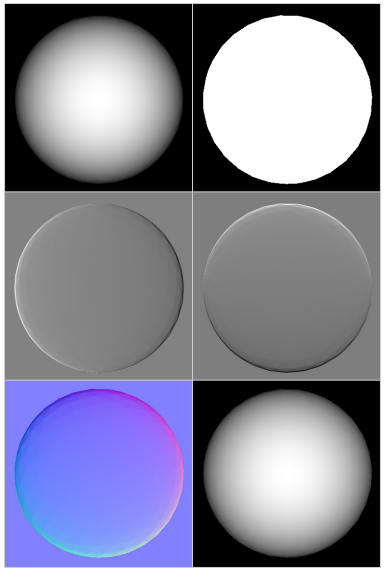

Figure 15: Illustration of the images that are computed during the presented shape from shading implementation. (a) provides the intensity image, (b) is the binary mask, (c) x derivative, (d) y derivative, (e) normal map, and (f) displacement map.

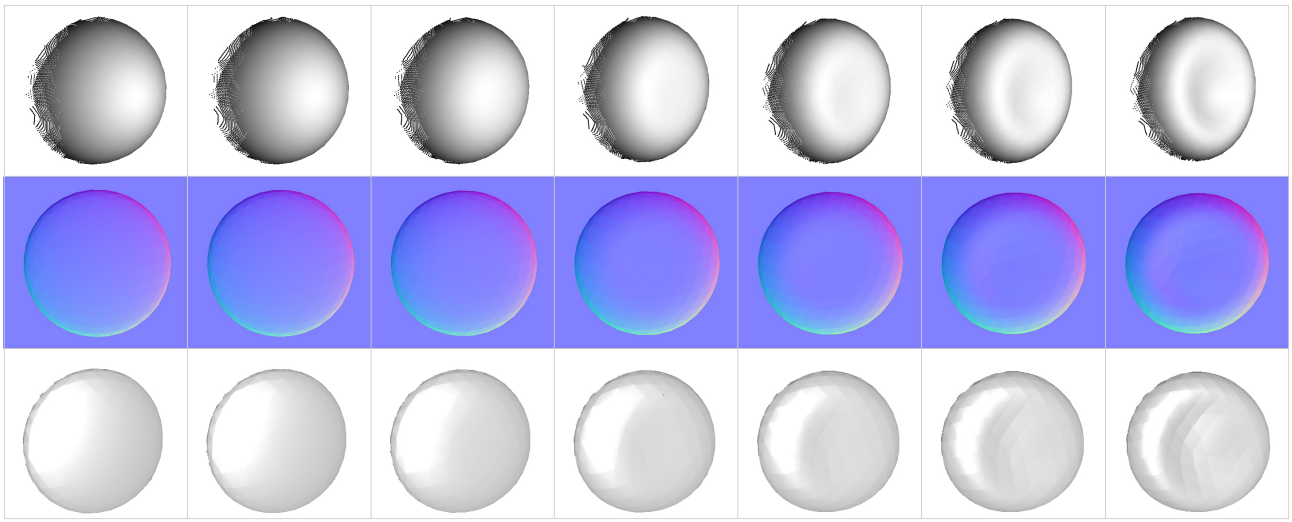

Figure 14: Overview of the image sequence that defines the synthetic deformation of a volumetric object using artificial force to cause the indentation. The (top) row illustrates the ground-truth of the deformation which has been obtained by the mathematical intersection of the surface during the ray-tracing procedure, the (middle) row illustrates the resulting normal map for the corresponding frame, and the (bottom) row illustrates the surface reconstruction of the surface. Important: the derivatives in this reconstruction were approximated using Central Finite Differences approximation filter.

6.2 Experimental Results

The experimental results of the implemented shape from shading technique were obtained using a simplified experimental setup for the lighting conditions, imaging device configuration, and the deformation imposed on the surface of the recorded object. For the experiment the Kinect2 was used as the recording device, mounted on a tripod and aimed at the deforming object at a distance of approximately 2.0[m] (just past the shortest depth reading measurements that the Kinect2 can provide). This provides the highest resolution of the object possible and ensures the highest quality of the depth image measurements (which is about at 2.0[m]).

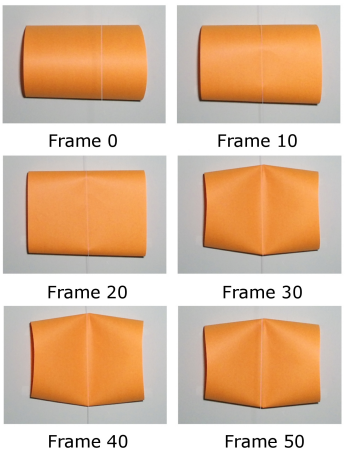

Figure 16: Illustration of the deformation sequence as seen by the color camera of the Kinec2. The experiment was executed as follows: (1) the string was pressed into the surface of the object to produce the deformation within the surface and (2) once the deformation was in place, it remained as a permanent fold within the surface of the object.

In the transition from performing the deformation analysis on the synthetic images to the real-world images, two modifications were added: (1) the color of the object was made distinct from the background to simplify the segmentation process, therefore a simple HSV color thresholding mask could be generated for the object, and (2) since the real-world imaging device contains noise, two different filters were applied to the intensity image (obtained by converting the RGB image to a gray-scale equivalent using the coefficients: $rc = 0.2989$, $gc = 0.5870$, $bc = 0.1140$). The first filter applied was a low-pass Gaussian filter, however this produced a poor result within the surface quality, therefore a simpler averaging filter (11 x 11) was applied to drastically improve the quality of the derivatives computed from the intensity image.

Using the real world images, the presented approach was tested using the input image sequence shown in Figure 16. From the HSV object segmentation applied, the erode and dilate morphological operations were applied using (3 x 3) and (5 x 5) ellipsoid structural elements. This drastically reduced the holes within the binary mask used to segment the orange object from the background for each frame. It was critical that this process was robust enough to work properly for each frame, otherwise the quality of the animation would suffer and the there would be a discontinuity within the deformation of the surface. The results of the final deformation result illustrating the intensity image, normal map, displacement map, and surface are shown in Figure 18.

The process of reconstructing an arbitrary markerless object using shape from shading is in general an ill-formed problem with respect to determining the exact depth of an object viewed from an monocular imaging device. While this is the case, we can still analyze the curvature of the surface based on the integrated depths obtained using the spiral-based integration technique. The images below in Figure 17 illustrate the numerical difference between the depth image (left) and the shape from shading result (right) .It is immediately noticeable that the noise level within the reconstructed surface is significantly less, however due to the inaccuracy of the surface reflectance properties, the apparent height of the center of the object has become skewed. This however validates that the respective surface curvature of the object has been accurately extracted from the image sequence.

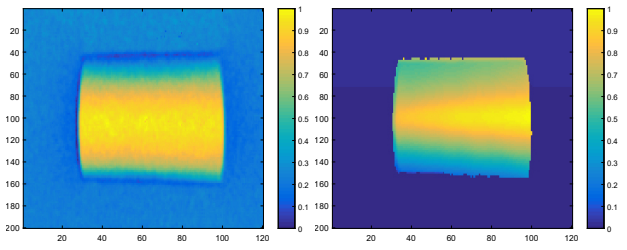

Figure 17: Numerical illustration of the height of the surface from the original depth image (left) and the reconstructed surface using the proposed technique (right). The noise within the depth image contributes to a significant variance within the surface.

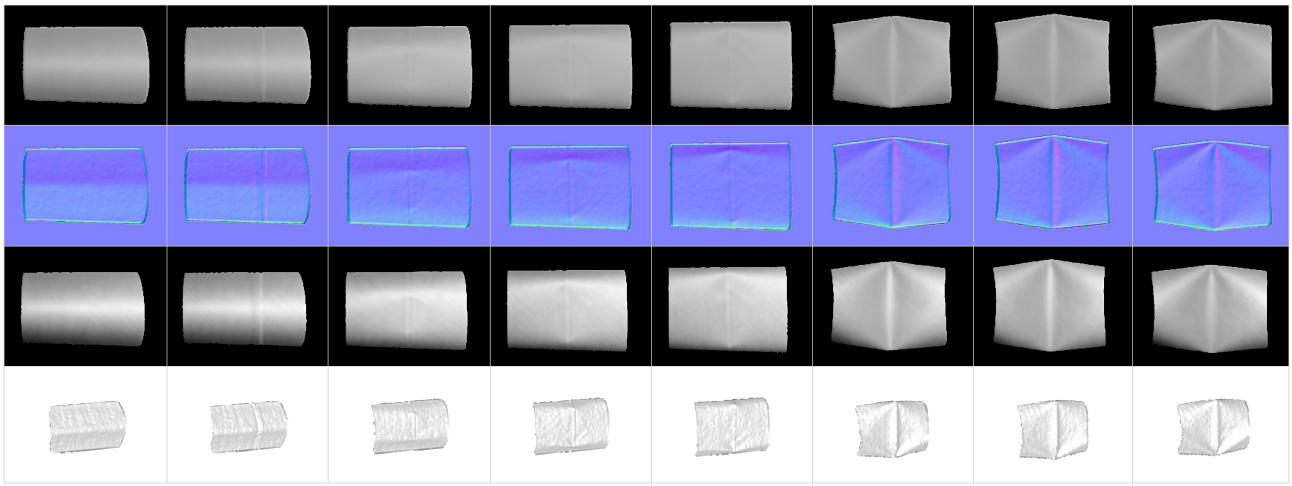

Figure 18: Overview of the image sequence that defines the real-world deformation of a volumetric object attached to a wall. The recorded sequence contains 60 total frames. During the deformation the string is pressed into the surface of the the volume causing a localized deformation of the surface.

7 Evaluation and Discussion

The presented shape from shading technique provides a viable method for observing and reconstructing surface deformations of both synthetic and real-world objects. While the synthetic results are more reliable due to the absence of noise, the real-world experiment also illustrates the potential for more accurate surface deformation analysis using digital image processing techniques. There are several aspects of the current implementation that could be improved including: (1) the SFS model used, which could account for a more dynamic light position and more robust formulation, (2) the approximation of the image derivative - even higher order Finite Differences could be used to reduce the error within these images, (3) the integration technique used to compute the depth of the surface. However, both the image sweeping technique and the spiral-based integration schemes were employed to generate the displacement map. The difference between these two techniques is significant and illustrated below within Figure 19.

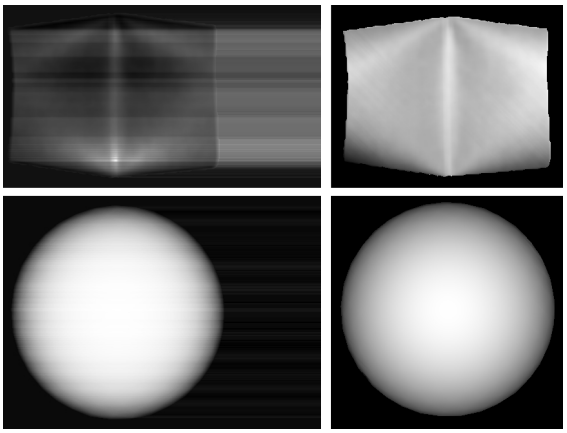

Figure 19: Improvement of the numerical integration technique used to generate the displacement map (or surface depths) of the deforming object. The sweep approach result is shown in the (left column) for the real (top) and synthetic images (bottom). The spiral-based integration technique (right column) provides drastic improvements for both the real and synthetic images. This improved the definition of the reconstructed surface considerably.

The improvement of the integration technique drastically improved the quality of the reconstructed surface due to the difference in the error accumulation in the images. For the sweep-based integration it is clear that the error within the height estimates is propagated to the right of the image (even for the synthetic image which has no noise). The spiral-based technique with the integration kernel previously presented drastically improves the height-map used to reconstruct the surface of the object for the final result.

An interesting application of the implemented SFS surface extraction technique is that it can be used to improve the visual fidelity of the depth image that is collected from the Kinect2. The proposed approach is used with the lower resolution of the color values provided for the colored depth image to obtain the image-based surface normals. These normals are then applied to the depth image. The image within Figure 20 illustrates the application of the surface curvature generated in the proposed technique (right) applied to the color point-cloud that is generated from the Kinect2 depth map [vs] the surface normals that are generated using the standard PCL k-neighborhood approach (left).

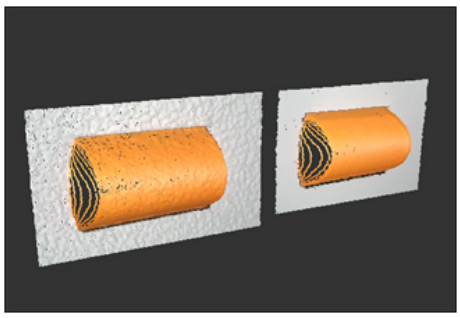

Figure 20: Application of the presented SFS surface curvature extraction process to the depth-cloud collected by the Kinect2. The application of the surface normals extracted from the color image that contains significantly less noise than the depth-image results in a surface of the object that appears much smoother.

The result of applying the SFS surface normals to the depth image increases the visual fidelity of the depth-cloud surface significantly. This indicates that the proposed method for approximating the surface curvature of the object is valid and can be used to at least improve depth-images when lighting conditions are ideal.

While many problems with the presented technique were identified and partially solved, there are still considerable challenges with implementing a SFS model extraction technique using monocular imaging device. This is because the problem is not well formed and can be unbounded without the correct assertions. Within this project, several assumptions were made to greatly simplify the process and ensure that the curvature of the surface of the deforming object can be closely monitored to observe localized deformations within the objects surface. One primary problem that has not been addressed within this work is the

concave/convex problem. When the surface gradient of the object is deforming, the surface normal does not provide enough information about the surface to define if there is a convex maximum or concave minimum. One way of addressing this problem is by doing critical-point analysis to determine if the curvature of the surface defines a local minimum or local maximum.

8 Conclusion

In this paper I have presented and implemented a simplified form of the Shape from Shading algorithm to perform surface deformation reconstructions from a sequence of images both from synthetic and real-world images. The reconstruction process is defined as: (1) the collection of intensity images, (2) the segmentation of the deforming object, (3) the computation of the derivative images using the Finite Differences approximation, (4) the approximation of surface normals and normal map generation, (5) the integration of the height differential field to obtain the displacement map, and finally (6) the reconstruction of the surfaces that define the objects deformation over time. This process was validated (with some constraints) using synthetic images and applied to real-world images collected with the Kinect2. Furthermore, the surface curvature estimations were applied to the depth images obtained from the Kienct2 to improve the visual fidelity of the point-clouds captured from the Kinect2.

References

Bichsel, M. and Pentland, A. P. (1992). A simple algorithm for shape from shading. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, pages 459–465.

Brooks, M. J. and Horn, B. (1985). Shape and source from shading. In *International Joint Conference on Artificial Intelligence*, pages 932–936.

Cournia, N. (2008). Creating height maps from normal maps. *Clemson University Slides*.

Dupuis, P. and Oliensis, J. (1992). Direct method for reconstructing shape from shading. In *IEEE Proceedings on Computer Vision and Pattern Recognition*, pages 453–458.

El, R. O., Rosman, G., Wetzler, A., Kimmel, R., and Bruckstein, A. M. (2015). Rgbd-fusion: Real-time high precision depth recovery. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5407–5416.

Frankot, R. and Chellapa, R. (1988). A method for enforcing integrability in shape from shading algorithms. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 10, pages 439–451.

Han, Y., Lee, J. Y., and Kweon, I. S. (2013). High quality shape from a single rgb-d image under uncalibrated natural illumination. In *2013 IEEE International Conference on Computer Vision*, pages 1617–1624.

Healey, G. and Jain, R. (1984). Depth recovery from surface normals. In *ICPR'84*, pages 894–896.

Horn, B. K. P. (1970). *Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from one View*. PhD thesis.

Horn, B. K. P. (1990). Height and gradient from shading. *International Journal of Computer Vision*, 5:37–75.

Wu, Z. and Li, L. (1988). A line integration based method for depth recovery from surface normals. In *Pattern Recognition, 1988., 9th International Conference on*, pages 591–595 vol.1.

Zhang, R., Tsai, P.-S., Cryer, J. E., and Shah, M. (1999). Shape-from-shading: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706.

9 Appendix

Implemented Application: The image below in Figure 21 provides an illustration of the implemented application. The application has been implemented in C++ using Qt, OpenCV, OpenGL, OpenMP, and the Kinect2 SDK within Visual Studio 2013.

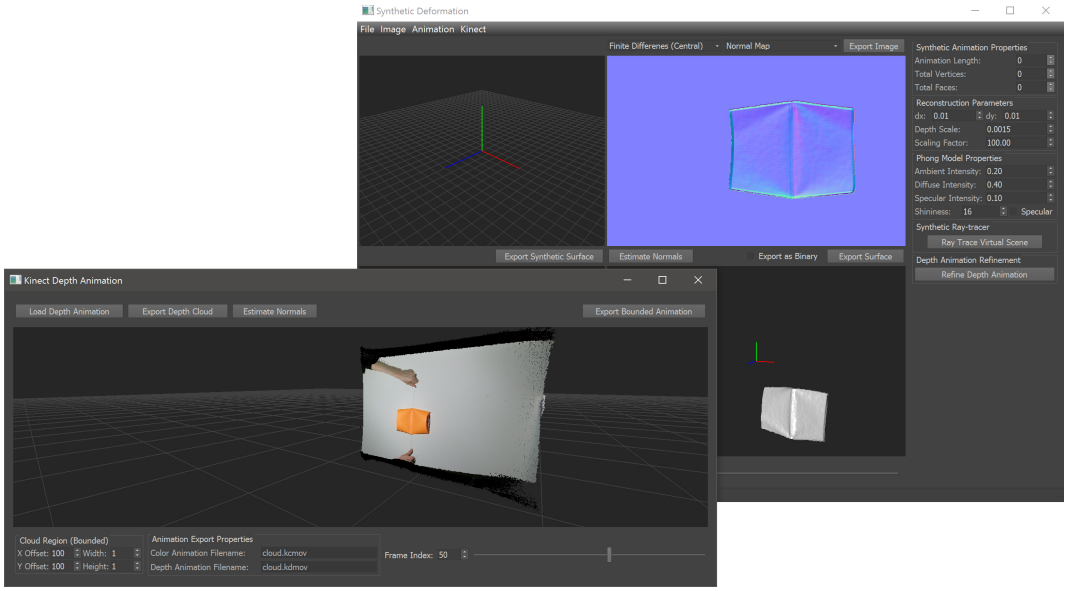

Figure 21: Screenshot of the implemented interactive surface reconstruction editor. The lower window illustrates the recorded depth image with color to illustrate the original real-world deformation. The larger window illustrates the main editor used for reconstructing the observed surface deformation where the normal (top-right) and surface (bottom-right) are shown within the applications viewports.

Experiment: The images below in Figure 22 illustrate the experiment that was performed to obtain the real-world image sequence. This experiment was simplified to eliminate visual interference and collision.

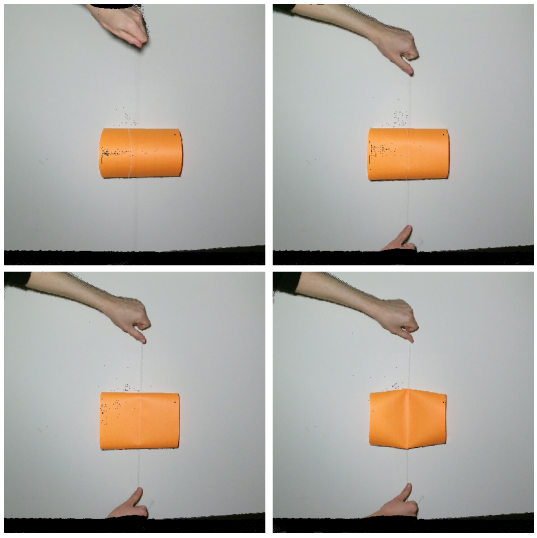

Figure 22: Illustration of the experiment from the full perspective of the imaging device. These images represent the colored depth image (or point-cloud) captured by the Kinect2.

Source Code: The source code for this project is too extensive to list as part of the appendix. The required source code for this project contains 80 C++ classes spread across 8 modules. All image processing techniques presented within this approach were implemented from scratch (using no libraries) with the exception of the OpenCV HSV segmentation, dilation, erosion, and Gaussian blurring, all other techniques for spatial convolution, intensity image manipulation, kernel application, average filtering, derivative computation using both Finite Differences and the Sobel filter, normal map generation, numerical integration, displacement map generation, and surface reconstruction are all implemented without the use of any external libraries. The total number of lines of C++ code implemented for this project (not including any library code) is: 397 + 591 + 128 + 58 + 460 + 144 + 432 + 335 + 49 + 388 + 409 + 43 + 43 + 95 + 54 + 388 + 44 + 340 + 163 + 287 + 44 + 302 + 44 + 103 + 335 + 90 + 834 + 165 + 75 + 55 + 39 + 39 + 44 + 292 + 80 + 67 + 989 + 277 + 169 + 70 + 145 + 227 + 60 + 59 + 59 + 57 + 68 + 62 + 55 + 67 + 55 + 95 + 108 + 78 + 159 + 102 + 160 + 115 + 74 + 249 + 48 + 57 + 65 + 115 + 47 + 65 + 251 + 512 + 83 + 165 + 220 + 230 + 146 + 690 + 970 + 715 + 68 + 431 + 769 + 887 + 902 + 161 + 80 + 55 + 57 + 64 + 67 + 102 + 147 + 59 + 59 + 143 + 126 + 431 + 35 = 20,037