KINECT CLOUD NORMALS: TOWARDS SURFACE ORIENTATION ESTIMATION

BY
PRATIK SANJEEV RUNGTA

THESIS
Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the Graduate College of the University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Adviser:
Professor Derek Hoiem
ABSTRACT

Since the early days of Computer Vision, we have explored what is possible in the realm of ‘Scene Understanding’. The advent of consumer-grade RGBD cameras has broadened the possibilities within this realm. The data they provide is able to serve as ground truth information or training data for a class of algorithms, which would otherwise be extremely difficult, if not impossible, to train. This thesis serves the purpose of gathering data from such a source, specifically, it demonstrates how to collect a dual pair of depth and RGB images of a multitude of scenes and an approach to determine surface normals from these images.

The goal of this endeavor is to provide a dataset of RGB images and surface normal estimates for each image so that the latter may serve as the ground truth for both training and evaluation of algorithms estimating surface normals from the RGB image alone.
To my father, for being the man he is.
ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Prof. Hoiem for his guidance, encouragement and patience. I am indebted to him for all of this and more.

I would like to thank my family, who have been supportive of me in every way imaginable. I will never be able to repay their kind words and encouragement. Words fail to describe the gratitude I feel towards my father; he is an incredible man who’s given me everything I’ve every needed, even if I didn’t know I needed it. My mother has given up herself for the sake of raising her children and I am truly grateful for her love and support. My sister, Avantika, has been a great listener and encouraged me to be myself in every situation imaginable. Her advice helped shape me in ways I did not imagine. My brother, though he doesn’t know it, is my ace in the hole.

Last, but certainly not least, I would like to thank all my friends for being the jovial and carefree souls they are. They amused, spoilt and supported me through my time with them. For this, I am grateful beyond measure.
# TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION .............................................. 1

CHAPTER 2 DATA COLLECTION ........................................... 3
  2.1 Kinect Hardware .................................................. 3
  2.2 Collection Framework ........................................... 4
  2.3 Dataset Compilation ............................................. 6

CHAPTER 3 DATA PROCESSING ........................................... 8
  3.1 Normal Computation ............................................. 8
  3.2 Method Justification ............................................ 10
  3.3 Benchmarks ...................................................... 11

CHAPTER 4 FUTURE WORK & CONCLUSIONS ......................... 14

APPENDIX A DATASET ACCESS METHODS ............................. 17
  A.1 Format .......................................................... 17
  A.2 Tools .......................................................... 18

APPENDIX B PROCESSING GROUND TRUTHS ....................... 19
  B.1 Planar Object .................................................. 19
  B.2 Cylindrical Object ............................................ 21
  B.3 Spherical Object ................................................ 25

REFERENCES .......................................................... 28
CHAPTER 1

INTRODUCTION

Since the early days of Vision, the community has strived to develop a wide range of algorithms under the banner of ‘Scene Understanding’. Over time, there have been many ideas which were not brought to fruition due to the lack of easily available sources of data needed to validate the initial insight. With the progress of technology, such instances of the lack of data holding back research are getting to be fewer and farther between.

One such case of the lack of data holding back research is the estimation of surface orientation from RGB images. There has been little research done within this sphere due to the difficulty in acquiring datasets for the training and validation of any algorithms. It should be noted here that, traditionally speaking, gathering depth related information from an image required reconstruction of scenes by taking pictures with multiple cameras or using expensive range cameras. Though the former is not technically unsolvable, it still poses a formidable barrier to entry. With the advent of consumer grade RGBD camera products, such as the Microsoft Kinect, this barrier to entry has eradicated.

Since the induction of this product, there has been a seeming gold-rush around exploring the boundaries of what it has to offer. This thesis was undertaken with the goal of using a Kinect to collect a large enough dataset (roughly 150 RGBD images), thereby facilitating the future exploration within the sphere of algorithms requiring such data, while focusing on calculating the ground truths which will be required for surface orientation estimators.

The remainder of this thesis is organized as follows:

- **Chapter 2** focusses on the data collection and compilation aspect of the thesis. It details the specifics pertaining the format, the test setup, etc.

- **Chapter 3** handles the subsequent processing of the collected data. It
details the surface normal computation, along with a few benchmarks justifying our choice of method.

- **Chapter 4** concludes the work of this thesis with closing remarks and explains the potential for future work.

- **Appendix A** deals with the additional tools and recommendations aimed at facilitating interaction with the dataset.

- **Appendix B** describes the process undertaken to benchmark the normal estimation techniques, including ground truth estimation and analysis.
CHAPTER 2

DATA COLLECTION

To begin a discussion of the dataset collection, the most appropriate place to start is by describing the Kinect itself. Hence, this chapter begins by describing the Kinect hardware, how it is used to capture data and finally the format and structure of the data collected.

2.1 Kinect Hardware

The Kinect is a device equipped with a RGB camera, an IR camera and a laser-based IR projector. The device is able to capture a combination of video streams at 30Hz.

The RGB camera acquires a 640x480 8-bit image for each frame, while the stereo pair of the IR camera and projector use structured light by projecting a grid of IR points and comparing the pattern formed by the subsequent diffraction gratings with precomputed values to estimate a 640x480 11-bit depth image. These two images overlap over a majority of the regions being captured, the small disparity occurs due to the different physical locations.

Figure 2.1: A Kinect without its outer frame – source:[1]
Figure 2.2: Sample Scene. Left: RGB Image; Right: Depth Image — the
darker the pixel, the closer it is, except for black which indicates no reading
of the two sensors — as is evident from Figure 2.1. These two images can
be calibrated and using the depth information from the IR image, we are
effectively presented with a 4 channel image containing the three standard
color channels, along with a depth channel. Acquiring a large set of such
images is one of the goal of this thesis.

2.2 Collection Framework

There has been extensive work done for the calibration and depth estimation,
including multiple open source frameworks and even an official SDK. Instead
of starting from scratch, I chose to rely upon one such stable open source
framework called OpenNI. It is a stable community driven project which
can successfully perform all the intermediary computation required. Figure
2.2 is an example of the output generated using the tools provided by the
framework.

These 4 channel images can be converted into alternative representation
of point cloud data which requires first and foremost the choice of a coordi-
nate frame. Figure 2.3 visualizes the coordinate frame we used. Our raw 4
channel images contain additional implicit information in the form of pixel
coordinates. Given the coordinates of a pixel, \((u, v)\) on the image, we know
its depth \(Z(u, v)\) from the data collected. Using this information, we calcu-
late two new parameter maps describing the X and Y coordinates using the
relations described below — the variables used in the relation are described
Figure 2.3: Coordinate Frame Choice [2]

in Table 2.1.

\[ X(u, v) = \left( \frac{u}{w} - c_x \right) \cdot f_x \cdot Z(u, v) \]  
\[ Y(u, v) = \left( c_y - \frac{v}{h} \right) \cdot f_y \cdot Z(u, v) \]

Table 2.1: Variable Explanation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w )</td>
<td>Width of Image</td>
<td>640px</td>
</tr>
<tr>
<td>( h )</td>
<td>Height of Image</td>
<td>480px</td>
</tr>
<tr>
<td>((c_x, c_y))</td>
<td>Principal point offset(^1)</td>
<td>(0.5, 0.5)</td>
</tr>
<tr>
<td>( f_x )</td>
<td>( x ) scale factor(^1)</td>
<td>1.1114666461944580e + 00</td>
</tr>
<tr>
<td>( f_y )</td>
<td>( y ) scale factor(^1)</td>
<td>8.3359998464584351e − 01</td>
</tr>
</tbody>
</table>

It should be noted here that, Equations 2.1 and 2.2 are conventionally written in the format of Equations 2.3 and 2.4. These two formats are equivalent, Table 2.2 gives the formulas for conversion between the two.

\[ X(u, v) = \frac{1}{f_x^p} \left( u - u_0 \right) \cdot Z(u, v) \]  
\[ Y(u, v) = \frac{1}{f_y^p} \left( v_0 - v \right) \cdot Z(u, v) \]

\(^1\)These four variables comprise the intrinsics of the depth camera — they are extracted from the OpenNI framework. For more information about Camera Models, refer to [3].
Table 2.2: Transformation Between Eq 2.1-2.2 and 2.3-2.4

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Transformation Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(u_0, v_0)$</td>
<td>Principal point offset</td>
<td>$(w \cdot c_x, h \cdot c_y)$</td>
</tr>
<tr>
<td>$f_x^p$</td>
<td>x focal length</td>
<td>$w/f_x$</td>
</tr>
<tr>
<td>$f_y^p$</td>
<td>y focal length</td>
<td>$h/f_y$</td>
</tr>
</tbody>
</table>

2.3 Dataset Compilation

Using these tools, we collected a dataset of 150 RGBD images and subsequently processed them using the techniques described in the upcoming chapters. Our current workflow is depicted in Figure 2.4. After the processing layer, we focus on the compilation of the dataset into an easy to access format. This section focusses on the compilation aspect of the thesis.

Figure 2.4: Method Overview

In the processing phase, for each RGBD image in the dataset, we compute an additional six channels. These are:

**Kinect Mask** The Kinect sensor provides depth images which contain noise in the form of a few pixels containing “NaN” values (black regions in Figure 2.2). This happens because of two main category of reasons. One of which is the different physical positions of the sensors of the depth (IR) camera and IR projector, this results in missing pixels at the time of calibration. The second is due to the Kinect’s susceptibility to noise because of reasons like specularities, diffraction gratings around object edges convoluting the estimates, etc. **Kinect Mask** is a logical
map, a True value indicates the sensor returned depth information for the particular pixel and vice versa.

**Coordinate Maps** Using Equations 2.1 and 2.2, we generate the $X$ and $Y$ coordinate maps.

**Normal Maps** Using the techniques described in Section 3.1, we calculate the surface orientation normals for each point. These are represented as three additional maps, one for each dimension.

Hence, after processing, for each of the images in the dataset we have 10 channels of information — RGB, XYZ, $N_x$, $N_y$, $N_z$ and the Kinect mask. After the processing, we compile these into a single Matlab workspace within which, the ten channels are stored as matrices, thereby permitting easy access and manipulation of the dataset. For more information about access methods and tools for dataset manipulation, please refer to Appendix A.
CHAPTER 3
DATA PROCESSING

This chapter deals with the necessary computation performed upon the collected dataset to estimate the surface normals. A sample result of this computation performed upon the raw data displayed in Figure 2.2 is displayed in Figure 3.1.

3.1 Normal Computation

There has been a lot of work done on extracting normals from range images, [4] provides a good survey of the various techniques developed for such scenarios. Our method — described below — is adapted from their recommendations.

The problem of normal computation, clearly stated, reads as follows: given a set of \( N \) points, \( P \) in 3D, compute the normal at each point. i.e. given, \( P = \{p_i| \text{where } p_i = (x_i, y_i, z_i)^T \text{ and } 1 \leq i \leq N\} \), compute the set of normals, \( \{n_i|1 \leq i \leq N\} \).

To do so, we compute a neighborhood matrix, \( Q \), for each point of the point

Figure 3.1: Estimated Unit Normal Map for Figure 2.2. Left-Middle-Right: X-Y-Z component maps of the normal vector.
cloud. The criterion for selecting points in the neighborhood is the euclidean distance from the initial point, the neighbors are thresholded to be within distance $r_n$ from the initial point. Equation 3.1 provides a mathematical definition of the neighborhood set $NQ_i$ for the point $p_i$.

$$NQ_i = \{q_j | q_j \in P \text{ and } \left\| p_i - q_j \right\|_2 \leq r_n \}$$  \hspace{1cm} (3.1)

This neighborhood matrix is represented as a matrix in the following way:

$$Q_i = [q_{i1}, ..., q_{ik}]^T$$

where $\forall j \in [1, k], q_{ij} \in NQ_i$ and $k = |NQ_i|$

To find the normal vector for this point, $n_i$, we find the plane to this neighborhood which minimizes the variance. The optimization relation which does so is:

$$n_i = \min_{n_i} \left\| (Q_i - \tilde{Q}_i) \cdot n_i \right\|$$  \hspace{1cm} (3.2)

where $\tilde{Q}_i = \frac{1}{k} \tilde{q}_i^T$ is a matrix containing the centroid vector, $\tilde{q}_i = \frac{1}{k} \sum_{j=1}^{k} q_{ij}$ in every row.

The solution to the above optimization equation, can be easily derived using Singular Value Decomposition. The right singular vector corresponding to the smallest eigenvalue of $(Q_i - \tilde{Q}_i)$ minimizes the objective, and thereby gives us the solution for the normal vector up to the sign.

Figure 3.2: Inconsistent EGI [5] \hspace{1cm} Figure 3.3: Consistent EGI [5]

This ambiguity of the normals’ sign is represented visually by Extended Gaussian Images(EGI) or normal spheres, which describe the orientations of all normals from the dataset. As our dataset is acquired from a single
viewpoint, an EGI for our scene should be present on only half the sphere. Without dealing with the orientation ambiguity, we are left with a distribution similar to figure 3.2. But, by orienting each normal towards the viewpoint, $v_p$, we obtain a consistent solution similar to figure 3.3. To orient the normals in this way, we need to pick the sign for $n_i$ which satisfies equation given below. \(^1\) (Rusu, 2009)

$$n_i \cdot (v_p - p_i) > 0 \quad (3.3)$$

### 3.2 Method Justification

To verify our method of normal computation, we collected RGBD images of a few basic geometrical objects — a sphere, a cylinder and a plane — to serve as ground truths. Using the techniques described in Appendix B, we estimated the true normals for these objects. This chapter examines the comparison of the normals estimated using our method versus a simpler scheme of normal computation (described below) versus the true normals.

#### 3.2.1 Quality Metric

For the purpose of comparison, we use the metric of accuracy defined in [4], i.e. the quality, $\gamma_i$, of the estimated normal vector $n_i$ is the absolute value of its normalized dot product with the ground truth normal vector $\hat{n}_i$. As reflected by 3.5, $\Gamma$ is the mean quality of the normal estimation over all the pixels in the image, and thereby provides a more complete measure of the quality of the estimation. It is 1.0 for a completely accurate estimate and 0.0 for an entirely incorrect estimate.

$$\gamma_i := \gamma(n_i, \hat{n}_i) = \frac{|n_i \cdot \hat{n}_i|}{\|n_i\| \cdot \|\hat{n}_i\|} \quad (3.4)$$

$$\Gamma = \frac{1}{n} \sum_{i=1}^{n} \gamma_i \quad (3.5)$$

\(^1\)In our case, due to the choice of coordinate frame described in Section 2.2, $v_p = [0, 0, 0]^T$. 

10
3.2.2 Naive Normal Estimation

We also define an additional naive method of normal estimation — which simply chooses the normal at a point to be the gradient at that point. Equation 3.6 provides a formal definition of it. We do this to provide context for comparison against our method.

\[
n(u, v) = \left( X(u+1, v) - X(u, v), Y(u+1, v) - Y(u, v), Z(u+1, v) - Z(u, v) \right)^T
\]

(3.6)

3.3 Benchmarks

Using \(\Gamma\) as the metric for comparison of the naive method defined in Equation 3.6 and the method described in Section 3.1, we benchmarked a few basic geometrical objects: a sphere, a cylinder and a plane. For the sake of brevity, we skip images depicting the normal maps in this chapter. ²

Table 3.1: Normal Estimation Benchmark

<table>
<thead>
<tr>
<th>Object</th>
<th>Naive (\Gamma)</th>
<th>SVD((r_n = 5\text{mm}))(^3)(\Gamma)</th>
<th>SVD ((r_n = 20\text{mm}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plane</td>
<td>0.2558</td>
<td>0.9497</td>
<td>0.9963</td>
</tr>
<tr>
<td>Cylinder</td>
<td>0.2641</td>
<td>0.9633</td>
<td>0.9777</td>
</tr>
<tr>
<td>Sphere</td>
<td>0.6228</td>
<td>0.8756</td>
<td>0.9472</td>
</tr>
</tbody>
</table>

As can be inferred from Table 3.1, the SVD method of Normal estimation provides a high degree of accuracy in all cases. The method also appears to be working better the larger the local neighborhood of the initial point. We investigated this further by plotting the \(\Gamma\) versus \(r_n\) graphs for each of these objects (represented in Figures 3.4 – 3.6). We found that upon increasing \(r_n\) beyond 20mm, the accuracy gained for the sphere and plane plateau and the gain in accuracy for the cylinder is almost negligible (it’s already over .98 at 20mm).

For these reasons, we chose to use the SVD method as our method of choice for computing the normals for the entire dataset, and limited \(r_n\) to 20mm.

---

² All the image, coordinate and normal maps for the benchmarks are available in Appendix B.
³ SVD Method refers to the method of Normal Computation described in Section 3.1
Figure 3.4: Plot of accuracy of normal computation, $\Gamma$ on the Y-axis against thresholding distance, $r_n$ (in m) on the X-axis for Planar ground truth benchmark.

Figure 3.5: Plot of accuracy of normal computation, $\Gamma$ on the Y-axis against thresholding distance, $r_n$ (in m) on the X-axis for Cylindrical ground truth benchmark.
Figure 3.6: Plot of accuracy of normal computation, $\Gamma$ on the Y-axis against thresholding distance, $r_n$ (in m) on the X-axis for Spherical ground truth benchmark.
CHAPTER 4

FUTURE WORK & CONCLUSIONS

The primary purpose of this thesis was to collect a dataset which could serve as the ground truth for algorithms attempting to estimate surface orientations from only the RGB components. Using the techniques described in this thesis, we were able to gather and process a dataset of \( \approx 150 \) images across a variety of scenes across hallways, labs, households, libraries, grocery stores and apartments. The diversity in this dataset accounts for a great variety of shape and color texture, and hence makes it a feasible choice to serve as the required ground truth for surface orientation estimations. Figure 4.1 is a montage of the RGB images collected.

Looking towards the future potential for the work performed in the thesis, beyond serving as the ground truth for future surface orientation estimation algorithms, below is a list of topics which can be explored:

**Data Smoothing** The data provided by Kinect Sensor contains holes within the depth channel, which reduces the value of the collected dataset. One potential field to explore is to develop methods to fill in these holes. We are currently working attempting two techniques to do this interpolation. The first is using Radial Basis Neural Network to do function interpolation using RGBD data from the local neighborhood of missing values and the second is using combination of image segmentation (Quickshift, in our case) and surface fitting (planar or quadratic). Another interesting avenue to explore is to treat the surface orientation estimation as a subclass of non-linear filtering by using Bilateral Filtering.

**Improving Normal Estimates** The surface orientation estimates using the technique described takes a thresholding knob when computing the covariance matrix: the radius over which to include neighbors. As a result of this, the technique is biased towards choosing more neighbors for ob-
Figure 4.1: Dataset Montage
jects that are closer to the camera. It will be interesting to explore if we make this knob dynamically increase for points farther away from the camera, so we have the same number of neighbors for the computation. Alternatively, as done in [5] we can add another thresholding knob to limit the number of neighbors used for the computation, regardless of the distance of the point. The relationship of the radius versus the number of neighbors will be an interesting point to explore as well.
APPENDIX A

DATASET ACCESS METHODS

This chapter details the dataset and corresponding tools bundled with this thesis. We describe the format of the stored data (both raw and processed) and then provide a summary of the tools implementing the techniques mentioned in this thesis.

A.1 Format

The dataset, in its raw form, comprises a folder entitled Data, which contains a collection of numbered folders. Each one of these folders contains the source for a single RGBD image. The RGB component is stored in an image file called scene.jpg and depth channel is stored within points.pcd.\(^1\)

After processing, the data from each 4-channel image is converted into 10 channels. To facilitate easy access, for each 4-channel image there is a separate Workspace file with the same title, which stores the 10 processed channels. Table A.1 provides a description of contents of a Workspace file — note that each of variables listed is a 640x480 matrix (except the image itself, which is a 640x480x3 matrix).

\(^1\)The ‘PCD’ file format is a well documented public file format, for more information about it, please refer to [6].

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>im</td>
<td>8 Bit RGB Image</td>
</tr>
<tr>
<td>kmap</td>
<td>Logical Mask indicating pixels with depth returned by the Kinect</td>
</tr>
<tr>
<td>(x^2)</td>
<td>(x) Coordinate Map</td>
</tr>
<tr>
<td>(nx^3)</td>
<td>(x) Component of Normal Vector</td>
</tr>
</tbody>
</table>
A.2 Tools

Apart from the collection itself, all the computation performed in this thesis was performed using Matlab. The functions and scripts used for the same are briefly described in Table A.2. The source-code for these is available with the dataset; please refer to the comments in the files themselves for usage information.

Table A.2: Programming Tools

<table>
<thead>
<tr>
<th>Filename</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>read_pcd.m</td>
<td>Function to extract the depth map from a PCD file in the dataset.</td>
</tr>
<tr>
<td>generate_xyz.m</td>
<td>Function which generates the X and Y coordinate maps and the Kinect Mask map for the provided depth map.</td>
</tr>
<tr>
<td>compute_norm.m</td>
<td>Function which computes the surface orientations using the technique described in Section 3.1.</td>
</tr>
<tr>
<td>read_dataset.m</td>
<td>Script which reads in the raw dataset and performs the necessary computation to generate the Workspace files described in Section A.1</td>
</tr>
</tbody>
</table>

3: x’ can be replaced with ‘y’ or ‘z’
3: nx’ can be replaced with ‘ny’ or ‘nz’
In this chapter, we describe the process we undertook to estimate the normals for the ground truth objects. We first describe the process we undertook to compute the true normals. Along with this, for each object, we also provide the necessary maps representing the computation of the normals using the other techniques mentioned earlier.

To begin, for each shape, we computed a parametrization of the object so as to facilitate the computation of the true normals. After this step, we defined the necessary mathematical relations, depending upon the geometry of object, to estimate the normal. In the subsections below, we describe this process for each object.

B.1 Planar Object

B.1.1 Parametrization

The first object we deal with is a plane (Figure B.1). For our parametrization of the plane, we choose the parametrization described in Equation B.1. Using the X, Y and Z maps for this object (scaled for the purpose of display in Figure B.2), we are able to estimate a Least Squares Fit of the parameters of the plane. Table B.1 lists the estimated parameters.

\[ a \cdot x + b \cdot y + c \cdot z + d = 0 \] (B.1)
Figure B.1: Plane RGB Image

Table B.1: Plane Parameters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.14827</td>
</tr>
<tr>
<td>b</td>
<td>0.233964</td>
</tr>
<tr>
<td>c</td>
<td>−0.960873</td>
</tr>
<tr>
<td>d</td>
<td>0.990107</td>
</tr>
</tbody>
</table>

Figure B.2: Scaled Coordinate Maps
B.1.2 True Normal Calculation

Based on our parametrization of the plane, the normal at all points will be \( n_i = (a, b, c)^T \). As this will be a constant value for all the points across the plane, we choose not to include the maps for this normal.

B.1.3 Other Normal Estimation

Using the algorithm described in Section 3.1 and Section 3.2.2, we compute the normals for the planar object. These are represented in Figure B.3 and B.4.

B.2 Cylindrical Object

B.2.1 Parametrization

The next object we deal with is a cylinder (Figure B.5). We can parametrize this object by noting that any point on its surface must be at a fixed distance,
Figure B.5: RGB image of Cylindrical Object

Table B.2: Cylinder Parameters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{ax}$</td>
<td>(0.0240257, -0.984968, 0.171056)</td>
</tr>
<tr>
<td>$P_{on}$</td>
<td>(-0.0695742, 0.128755, 0.777683)</td>
</tr>
<tr>
<td>$r_c$</td>
<td>0.0631356</td>
</tr>
</tbody>
</table>

$r_c$ from the point closest to itself on the axis of the cylinder. Using this fact, we have two degrees of freedom to completely describe a unique point, $P$, on the surface of the cylinder — the first is point on the axis closest to $P$, which we refer to as $P_{closest}$ and the second is rotational freedom of the point around $P_{closest}$. We represent this mathematically in Equation B.2. We also note that, $P_{closest}$ can be defined with as any point lying along the axis of the cylinder, so we use represent it using the direction vector of the axis, $V_{ax}$, and a point lying on the axis, $P_{on}$ — as shown in Equation B.3.

So to completely describe a unique cylinder, we have to define the three variables: $V_{ax}$, $P_{on}$ and $r_c$. Using the X, Y and Z maps of the cylindrical object — scaled for the purpose of display in Figure B.6 — we used an iterative RANSAC search to estimate these three unique parameters. Table B.2 lists the estimated parameters.

\[ \left\| P - P_{closest} \right\| = r_c \quad \text{(B.2)} \]

\[ P_{closest} = P_{on} + \lambda \cdot V_{ax} \quad \text{(B.3)} \]
B.2.2 True Normal Calculation

Based on our parametrization of the plane, the normal at a point \( P \) can be calculated by subtracting the component of \( P - P_{on} \) on the axis from the position vector \( P \). This is described in Equation B.4 and visualized in Figure B.7.

\[
n_p = P - \frac{1}{\|V_{ax}\|} \left( P - P_{on} \right) \cdot V_{ax}^r
\]  

(B.4)

B.2.3 Other Normal Estimation

Using the algorithm described in Section 3.1 and Section 3.2.2, we compute the normals for the planar object. These are represented in Figure B.8 and B.9.
Figure B.8: Scaled SVD Normal Maps

Figure B.9: Scaled Naive Normal Maps
B.3 Spherical Object

B.3.1 Parametrization

The final object we deal with is a sphere (Figure B.10). We can parametrize this object by noting that any point on its surface must be at a fixed distance, \( r_s \) its center, \( C_s \). So to completely describe a unique cylinder, we have to define the two variables: \( r_s \) and \( C_s \). Using the X, Y and Z maps of the spherical object — scaled for the purpose of display in Figure B.11 — we used a Least Squares formulation to estimate the two unique parameters. Table B.3 lists the estimated parameters.

\[
\begin{array}{|c|c|}
\hline
\text{Variable} & \text{Value} \\
\hline
C_s & (32.4814, -39.3104, 765.5255) \\
r_c & 78.4926 \\
\hline
\end{array}
\]

B.3.2 True Normal Calculation

The normal, \( n_p \) at a point \( P \) can be calculated, simply by calculating the unit vector from the center of the sphere to \( P \). This is described in Equation B.5 and visualized in Figure B.12.

\[
n_p = \frac{1}{\|P - C_s\|}(P - C_s)
\]
Figure B.11: Scaled Coordinate Maps

Figure B.12: Scaled True Normal Maps
B.3.3 Other Normal Estimation

Using the algorithm described in Section 3.1 and Section 3.2.2, we compute the normals for the planar object. These are represented in Figure B.13 and B.14.
REFERENCES


