Technische Universität Kaiserslautern

Master Thesis

Improved Normal Inference from Calibrated Illuminated RGBD Images

Author:
Jingyuan Sha

Supervisor:
Prof. Dr. Didier Stricker
M. Sc. Torben Fetzer

Augmented Vision
German Research Center for Artificial Intelligence

June 25, 2022

Declaration of Authorship

I, Jingyuan Sha, declare that this thesis titled, "Improved Normal Inference from Calibrated Illuminated RGBD Images" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Abstract

Faculty Name
German Research Center for Artificial Intelligence

Master of Science

Improved Normal Inference from Calibrated Illuminated RGBD Images

by Jingyuan SHA

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor…

Contents

List of Figures

List of Tables

List of Abbreviations

LAH	**L**ist **A**bbreviations **H**ere
WSF	**W**hat (it) **S**tands **F**or

Physical Constants

Speed of Light $c_0 = 2.997\,924\,58 \times 10^8\,\mathrm{m\,s^{-1}}$ (exact)

List of Symbols

a	distance	m
P	power	W ($\mathrm{J\,s^{-1}}$)
ω	angular frequency	rad

To ...

Chapter 1

Introduction

Surface normal is an important property of a surface with many applications, like surface reconstruction, shadings generation and other visual effects. The quality of the surface normal is directly impact the performances of the techniques mentioned above. However, especially in the task of real-world object digitalization, the surface is usually hardly to be mathematically described in equations due to the elaborate details on the objects. Instead, it is common to use a group of points to describe the object surface, which is a memory economical solution and also can be easier measured by 3D scanners. The working principle of scanners are various due to the application scenarios, which consequently produce different point cloud structure. For the scanners without positions recording, the point cloud acquired after scanning is unstructured. In this case, every 3D point can be captured by different capture position, and neighbors are not defined by capture time, which increase the difficulty and computation for the neighbor based normal inference approaches. Furthermore, since the lack of inherent structure, the normal can hardly be inferred by the parallel approaches. To acquire the structured point cloud, a depth camera can be used for data collection. It captures the RGB-D images for the object, which includes the standard RGB image with depth information of each pixel. After camera calibration, the corresponding point cloud can be calculated based on the depth map and camera matrix. The advantages are, a structured point cloud is mapped directly based on the same capture position from a 2D depth map, the neighbor information of each point is identical to the corresponding pixel in the 2D depth map, which reduced a huge computational works. Unfortunately, the depth maps captured by the sensors are only semi-dense, which is mainly cause by optical noises and the reflections in dark and shinny areas. Consequently, it raises the challenges to the neighbor based approaches.

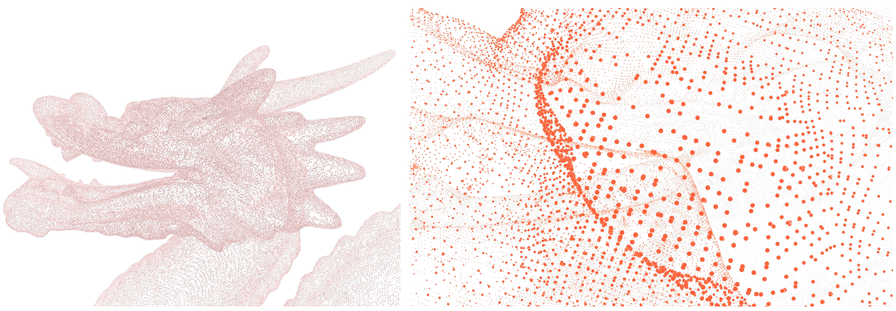


FIGURE 1.1: The point cloud representation of a dragon model. The right image shows the zoom in detail of the same model.

In order to acquire the point cloud of an object, a scanner is used to capture a set of scenes to depict the 3D object into a set of 2D images. Then process those images to reconstruct the object in the 3D space as a point cloud. When capture the images, the scanner can capture the RGB images as well as the corresponding depth map, i.e the RGB-D image. Using a scanner to project the 3D object to 2D images has multiple advantages. The pixels captures the RGB-D value of the points on the object surface and also the inherent structure of neighborhood, which makes the neighbors of the points much easier to be computed. Furthermore, it can also produce an illumination with determined pattern on the object, which further helps the inference of surface normal.

One of the desired direction is the 3d shape measurement. It is required to reconstuct the surface normal of the objects from images, which can be applied for tasks like 3D inspection, robotic manipulations and so on. The common way of 3D object measurement uses 3D scanner to capture the RGB image with corresponding depth map. Then convert the depth to structured point cloud with known intrinsic calibration parameters. Then based on these information infer the surface normal. The objects can be further illuminated by a source light with different patterns

1.1 Challenges of Normal Inference

However, due to the lack of accuracy of the sensors, the surface normal inference from depth maps/point clouds has many challenges. Recently, deep learning-based methods are used in computer vision tasks such as image segmentation, image inpainting, and depth density enhancement. In these methods, multiple architectures have proposed with an upsampling section. In this case, the output can be designed to have the same shape as the input or slightly smaller. Nevertheless, the resolution is similar to the input image. The normal Inference task has a similar pipeline compared to these tasks.

On the other hand, the point cloud data provided by Kinect or similar RGB-D, and LiDAR sensors are only semi-dense. A huge amount of missing pixels distribute all over the images, and some of the regions leave complete empty holes. This situation imposes another challenge on normal inference.

The depth image is incomplete, however, the depth sensor is able to capture grayscale texture images, which are typically fully dense due to their passive nature. Furthermore, if the texture image is already illuminated by strong directional light of a video projector, whose position is known, then there should exist theoretical relations between light direction, normal direction, and grayscale image. Then the normal can be inferred better using the given image information and depth map. Based on the grayscale image and corresponding semi-dense depth maps, a CNN model can be designed to infer the normal map, which can give more density and robust results comparing to the standard algorithm.

The normal map is usually saved as an RGB image which maps the X, Y, Z axes of the normal to R, G, B channels. Figure 1.4 shows the normal map of some basic geometry objects with RGB images.

1.2 Main Works of this thesis

In this thesis, we found a solution for the problems mentioned above and proposed a novel deep learning architecture for surface normal inference. A network named Albedo Gated Normal Inference Network(AlGaN) is proposed to infer normal given the corresponding depth map. The architecture of AlGaN involves a two-stage CNN. The first stage infers the normal map using point cloud as input. If the light source

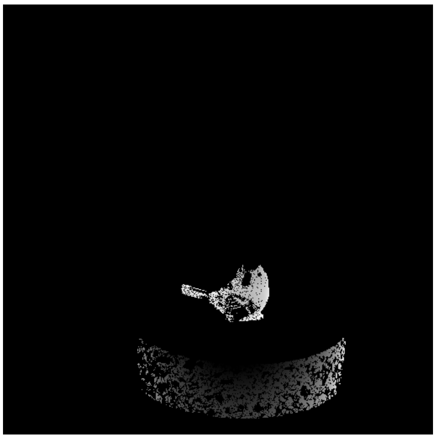


FIGURE 1.2: A captured depth map via infrared sensors. Pixels that far away represent by light colors, otherwise by dark colors. The black dots are the depths that failed to be detected.

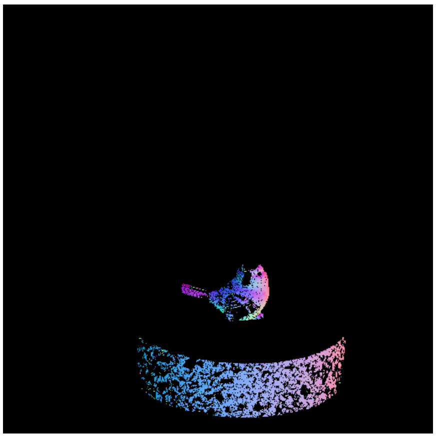


FIGURE 1.3: Semi-dense Normal Map calculated from depth map using a standard method

position and grayscale image can be provided further, the second stage infers the albedo altogether with the normals from the first stage. In addition, the loss function is based on least-square error and Lambertian reflection.

With the help of synthetic data in Unity, a dataset is created for CNN model training. It can provide accurate ground truth for training work, which real data is usually not provided. The results of this dataset show that our AlGaN model performs also well on real data captured by Infrared cameras. The trained Normal models achieve a remarkably better prediction accuracy at a low computational cost compared to the standard approaches for semi-dense point clouds.

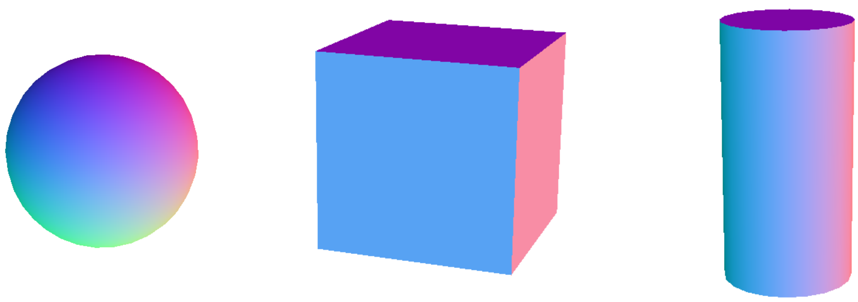


Figure 1.4: The surface normal stored in an RGB image where X, Y, and Z axes are mapping to R,G,B channels.

Chapter 2

Related Work

2.1 Standard Methods

Standard methods compute normals from the point cloud using neighboring information in image space or from a single grayscale image using use Shape from Shading Horn, 2004. The first method assumes that the neighbors of the points locate on the same plane. This method performs well with a well-chosen window size. However, the drawbacks are that the algorithm is highly noise sensitive. It is weak in handling missing pixels, which is a common issue in the input data. The second method depends on the correct information about the light source. Errors may occur in regions with inter-reflections in the 3D measurement.

2.2 Deep Learning based Method

Recently, deep learning based method Redmon and Farhadi, 2018, Tan, Pang, and Le, 2019 achieved a great succeed for image processing. These network architectures use a batch of RGB/Grayscale images as input and usually employ for classification problems. Usually, the images are convoluted with a convolutional layer and downsampling with pooling layers. The outputs of the network consist of a single value to represent the index of the corresponding class Tan, Pang, and Le, 2019, or with a set of values to represent the position of bounding boxes.Redmon and Farhadi, 2018.

However, in many other vision tasks, like normal map inference, the output is demanded as the same shape as the input. Instead of predicting one or several classes for the whole input matrix, the class for each pixel requires for prediction. In this case, the traditional network architecture is not suitable anymore.

From PCA for estimation to recently deep learning based methods, the task of surface normal inference is also been well studied in last several decades.

2.3 Sparse Input processing

The depth map is supposed to be dense. Therefore, how to accept sparse depth map as input in CNNs is one of the most important problem. Some trivial solutions like median filters are good enough, if the missing pixels are sparse enough, however, for the case of huge missing holes in the depth map, it produces just a paltry result. Thus a reasonable guess is required for missing areas. Generally, it can be solved as image inpainting problems,Yang, Kim, and Park, 2012,Qi et al., 2013.

Some deep learning based method for image inpainting also achieved quite good performance for the hole mending task. Notably, in 2016, Oord et al. Oord et al., 2016 proposed a gated activation unit for a CNN model,

$$\mathbf{y} = \tanh(W_{k,f} * \mathbf{x}) \odot \sigma(W_{k,g} * \mathbf{x})$$

to substitute the standard activation layer, where σ is the sigmoid function, which constricts the output value of second part between $[0, 1]$. The function is inspired by Long Short-Term Memory (LSTM) Hochreiter and Schmidhuber, 1997 and Rated Recurrent Unit (GRU).Cho et al., 2014 It is originally used for learning complex interactions as LSTM gates does. In 2018, Yu et al. Yu et al., 2018 employed same function for free-form image inpainting, which can be used to learn mask automatically from image it self.

Different to aforementioned approaches, Knutsson et al. in 2005 introduced normalized convolution Knutsson and Westin, 1993 dealing with missing sample case for convolution operation, which aims to reconstruct the missing pixels from the sparse output sensed by cameras, which particularly considered the confidence of each interpolated pixels, since it provides the trustworthiness of the predicted value. The higher the reliability of the value inference, the better the model shape reconstruction.

In 2018, Eldesokey et al. Eldesokey, Felsberg, and Khan, 2020a applied normalized convolution in CNN as normalized convolution layer that takes both sparse depth map and a binary confidence map as input to perform scene depth completion. In 2020, Eldesokey et al. Eldesokey et al., 2020 focus on modeling the uncertainty of depth data instead of assuming binary input confidence.

Guided method Hua and Gong, 2018 requires addition information like RGB image or the certainty map of depth map, and fuse them together to predict the dense depth map.

2.4 Normal Inference

Usually, we based on point cloud, depth map or RGB/Grayscale image of the objects or scenes to inference the normals.

Traditional methods evaluate normals based on neighbor information of point cloud or depth map. In 2012, Holzer et al. Holzer et al., 2012 proposed method to calculate normal from covariance matrices. This method use integral image as input, which is able to run algorithm in a high frame speed. They smooth the depth data in order to handle the noise of depth image. The drawbacks are, as mentioned in the paper, the normals error go up when point depths change severely. In 2013, Fouhey et al. Fouhey, Gupta, and Hebert, 2013 proposed a method constructing a over-determined function systems to predict normals and solving it by algebra methods. Similarly, this approach gives a quick but coarse normal inference.

Recently, CNN based methods improve the performance of image processing to a brand new stage. In 2014, Eigen et al.Eigen, Puhrsch, and Fergus, 2014 proposed a method predicting depth map directly from RGB image using CNN. In this case, no depth map is required. In 2016, Laina et al. Laina et al., 2016 proposed a deeper network based on ResNet He et al., 2015 with a well designed upsamling part. In 2018, Qi et al. proposed GeoNetQi et al., 2018, it integrates both algebra method and also CNN method to inference depthmap based on Laina et al., 2016 Fouhey, Gupta, and Hebert, 2013.

It is worth to noticed that, the output of normal inference CNN model is not one or severl labels but an entire image or normal map with same size. Recently, Ronneberger et al proposed an architecture called UNet Ronneberger, Fischer, and Brox, 2015 for biomedical image segmentations. The architecture is shown in Figure **??**.The first half network is a usual classification convolutional network, the second half replace the pooling layers and traditional fc layers in the traditional CNNs to upsampling layers, thus in the end of the second half, the output is able to back to the input size. The proposed network can successfully assigned each pixel a class for segmentation tasks. Under this symmetric network, an input image is downsampled 3 times and upsampled 3 times. Output image has exactly the same size as input image. The downsampling and upsampling both have large number of feature channels, which guarantee the network propagates the information to higher resolution layers.

In some case, the input is unstructured point cloud which can not be fed into a CNN entirely. Thus, a challenge task connect to the deep learning is the input format. Since different point clouds have different sizes. In 2018, Ben-Shabat et al. Ben-Shabat, Lindenbaum, and Fischer, 2019 presented Nesti-Net. It predicts the normal point by point with the help of neighbor points. It fixed the distance of considering neighbors to provide an unified input for CNN. In 2021, Zhou et al. Zhou et al., 2021 presents a method considering overlapping of different patches (a group of neighboring points) as input to evaluate normals.

Chapter 3

Approaches

In three dimension geometry, a surface normal at the point P is a vector n perpendicular to the tangent plane of the surface at point P. The length of a normal is usually one, with a sign to represent the sides (interior or exterior).

3.1 Neighbor based surface normal estimation

For a point P in a surface, its normal n is a vector of the tangent plane Π of the surface at this point P. As long as find k vectors $\mathbf{v}_1, ..., \mathbf{v}_k \in \mathbb{R}^3$ on the tangent plane, the normal can be calculated immediately based on equation $v \cdot \mathbf{n} = 0$

Let $P_1, ..., P_k \in \mathbb{R}^3$ are k neighbors of point P in the point cloud. In order to find the normal n, we can assume the neighbor points and P are in the same tangent plane. Then

$$\mathbf{v}_i = P_i - P \quad \text{for } 1 \leq i \leq k \tag{3.1}$$

are k vectors on the tangent plane. Since they all perpendicular to the normal n, we have

$$\mathbf{v}_i \cdot \mathbf{n} = 0 \quad \text{for } 1 \leq i \leq k \tag{3.2}$$

The equation system can be further simplified as

$$M \cdot \mathbf{n} = 0 \tag{3.3}$$

where $M \in \mathbb{R}^{k \times 3}$ denotes the matrix constructed by $\mathbf{v}_i$. In order to avoid trivial solution, one more constraint should be added

$$\|\mathbf{n}_{3\times 1}\|_2^2 = 1$$

, which also let the normal to be a unit vector.

In order to calculate a valid normal, 3 points are required at least. For the sake of robust, more points can be used to reduce the measuring error. In this case, the equation system is over-determined, then the equation system mentioned above can be converted to follow optimization problem

$$\begin{aligned} \min \quad & \|M\mathbf{n}\|^2 \\ \text{s.t.} \quad & \|\mathbf{n}\|^2 = 1 \end{aligned} \tag{3.4}$$

which can be solved by singular value decomposition(SVD). Let the decomposition of $M = U\Sigma V^T$, The solution i.e. normal is the last column of V.

At last, all the normals should point ot view point S, thus the direction of a normal should be inverted if

$$\mathbf{n} \cdot (P - S) > 0 \tag{3.5}$$

3.2 Gated Convolution neural network for surface normal estimation

The standard convolution layer goes like this

$$O = \Sigma\Sigma W \cdot I \tag{3.6}$$

each filter is applied as a sliding window to walk through the whole matrix and calculates the output matrix. Every entry in the matrix counts in the operation. This is reasonable for image processing task with full-dense input, since no missing pixels exist. However, the depth-map and point cloud is only semi-dense. The valid and invalid pixels will be treated equally if we still perform standard convolution layers. Since the aim of the network is not learning the pattern of noise, but the noise with eternally changing patterns will confuse the network, and it fails the normal inference, a mask is required to distinguish two kinds of pixels.

Eldesokey, Felsberg, and Khan, 2020b use binary mask to indicate valid pixels, and further use normalized convolution to predict the output. The normalized convolution is shown as follows

$$O(x,y) = \begin{cases} \dfrac{\Sigma_i^k \Sigma_j^k W(i,j) \cdot I(x-i, y-j) \cdot M(x-i, y-j)}{\Sigma_i^k \Sigma_j^k W(i,j) \cdot M(x-i, y-j)}, & \text{if } \sum_i^k \sum_j^k M(i,j) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

where k is the kernel size, (x, y) is the position in input, (i, j) is the displacement in kenel, M is the corresponding mask. A binary mask uses 1 to indicate valid pixels and 0 otherwise. $\odot$ denotes element-wise multiplication.

Normalized convolution layer added the weight to the mask. However, a initialization for the mask is still required, and the propagation of the mask remain a tricky task.

3.2.1 Gated Convolution

Yu et al., 2018 proposed gated convolution using for image inpainting task.

The structure is shown in Figure 3.1. Instead of using a mask as input to indicate valid pixels, it employs a standard convolution layers to learn this mask directly from data. The valid pixels are then activated by a Sigmoid function. Then it imply element-wise multiplication with the feature map. Formally, the gated convolution is described as follows, the layer with input size (N, C_{in}, H, W) and output size $(N, C_{out}, H_{out}, W_{out})$:

$$o(N_i, C_{o_j}) = \sigma\left(\sum_{k=0}^{C_{in}-1} w_g(C_{o_j}, k) \star i(N_i, k) + b_g(C_{o_j})\right) * \phi\left(\sum_{k=0}^{C_{in}-1} w_f(C_{o_j}, k) \star i(N_i, k) + b_f(C_{o_j})\right) \tag{3.8}$$

where ϕ is LeakyReLU function, σ is sigmoid function, thus the output values are in range $[0, 1]$, $\star$ is the valid 2D cross-correlation operator, N is batch size, C denotes a number of channels, H is a height of input planes in pixels, and W is width in pixels, $w(C_{o_j}, k)$ denotes the weight of j-th output channel corresponding k-th input channel, $i(N_i, k)$denotes the input of i-th batch corresponding k-th input channel, $b(C_{o_j})$ denotes the bias of j-th output channel.

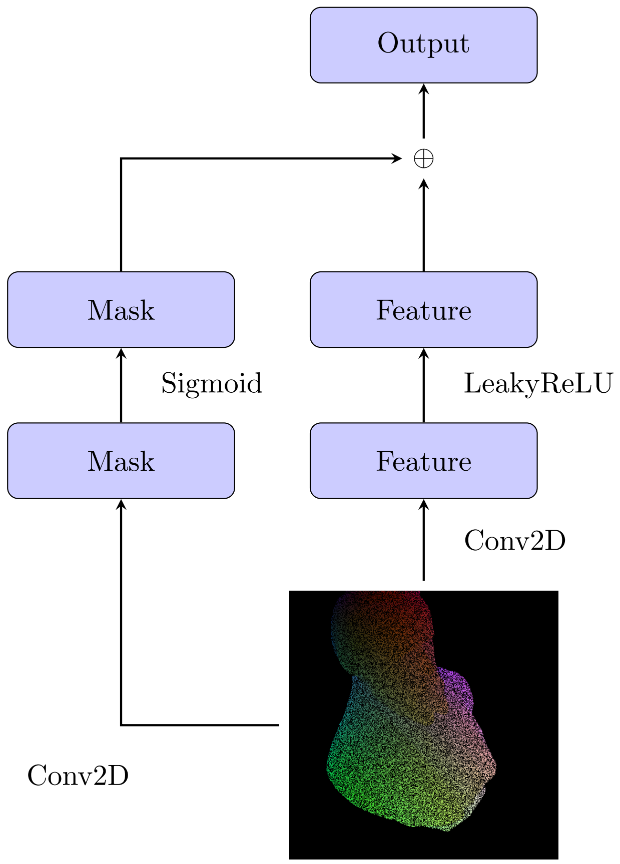


FIGURE 3.1: Gated Convolution Layer, where $\oplus$ denotes element-wise multiplication.

3.2.2 Architecture

Based on the implementation mentioned above, the architecture roughly follows on UNet proposed by Ronneberger, Fischer, and Brox, 2015, as shown in Figure 3.2.

Instead of using pooling layers for down/up samplings, gated convolution layer with stride $(2, 2)$ is used. The gated convolution using Sigmoid function for gating layer and LeakyReLU function for feature layer. All the layers are gated convolution layer with the exception of last two layer, which instead uses standard convolution layer to scale the output in range $[-1, 1]$. Other than the last two layers which use 1×1 kernels, all the gated convolution layer use 3×3 kernels with 1×1 padding.

The input is 3D vertex with size $512 \times 512 \times 3$, and output is $512 \times 512 \times 3$ normal map, which has the same resolution. There are 3 times downsamplings, each scale with 3 gated convolution layers, the third layer has stride-2. The upsampling part interpolate the feature maps 3 times with 1 gated convolution layer in each scale.

It keeps the skip connection in UNet to remain the fine detail features.

3.2.3 Loss Function

Perceptual Loss

Johnson, Alahi, and Fei-Fei, 2016 proposed perceptual loss.

L2 Loss

The loss function is based on mean square error is described as follows:

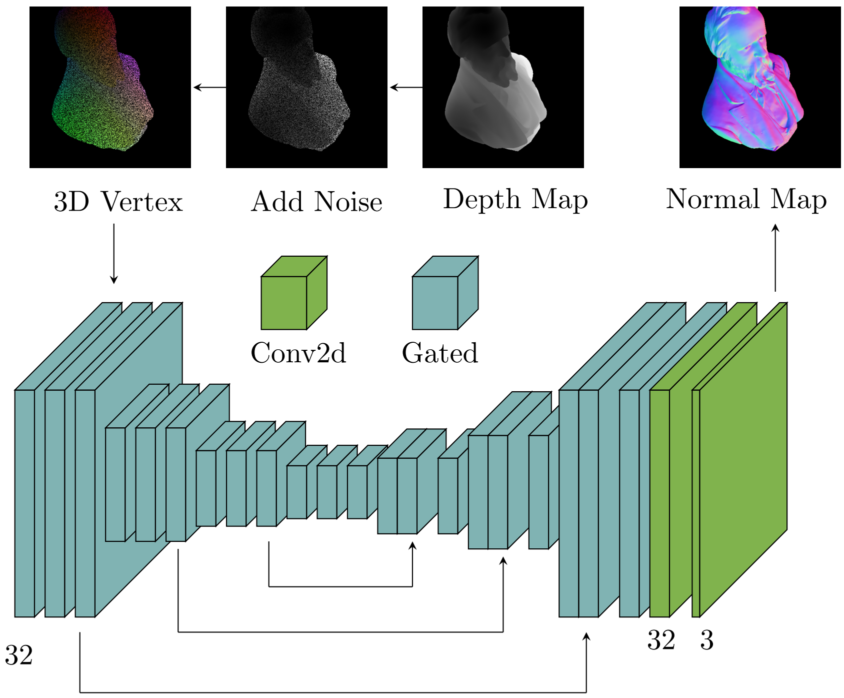


FIGURE 3.2: Basic Normal Neural Network model based on Gated Convolution layer and UNet architecture.

$$
\begin{aligned}
l(x,y) = L \quad &= \{l_1, ..., l_N\}^T \\
l_n \quad &= mean(Mask_{ol}(x_n - y_n)^2 \cdot p + (x_n - y_n)^2 \cdot Mask_{nol})
\end{aligned}
\tag{3.9}
$$

where x is input, y is target, N is the batch size. $Mask_{ol}$ is the mask for the outlier, p is the penalty of the outlier, it is set as 1.4.

3.3 Guided normal inference using GCNN

3.3.1 Image Guided normal inference

The normal inference can be guided by a RGB or gray-scale image, since the image is captured by passive method, it is fully-dense comparing to depth map, hence provides a complete view of the scene. The architecture is shown in Figure 3.3. The upper branch is the similar with GCNN model but with 4 additional concatenate layers, furthermore, 1 gated convolution layer is added before concatenate with image branch. The image branch takes a single grayscale image as input, then 3 times downsamplins with 3 standard layers in each scale. In the upsampling part, the feature map upsampled 3 times and concatenate with the last layer in the downsampling part before interpolation.

3.3.2 Add the light information

stores for every pixel the direction of incoming light. find a mapping H, such that $l_{in} = H(v, l_s)$ for pixel v, l_s is the position of light source, l_{in} is direction of incoming light of pixel v. Iterate all the pixels, we get the light direction map L.

Let I denotes image, N denotes normal map, L denotes light direction map.

lambertian reflection

$$I = \rho N * L$$

where $*$ denotes scalar product, the albedo rho can be computed by

$$\rho = \frac{I}{N * L}$$

in NN, it is

$$\rho = conv2d(\frac{I}{N * L})$$

The corresponding loss term is

$$\|I - \rho(N * L)\|_{F_2}$$

in this case, the normal is supposed to be more crispy than the previous implement.

In a first step, we should compute initial albedos from the predicted normals and check if everything is correct. (Normals need to be in spatial space again, not in tangent space, and we need to use cameras extrinsics R and t from the data file to triangulate the vertex maps correctly)...

$$\rho = \frac{I}{N * L}$$

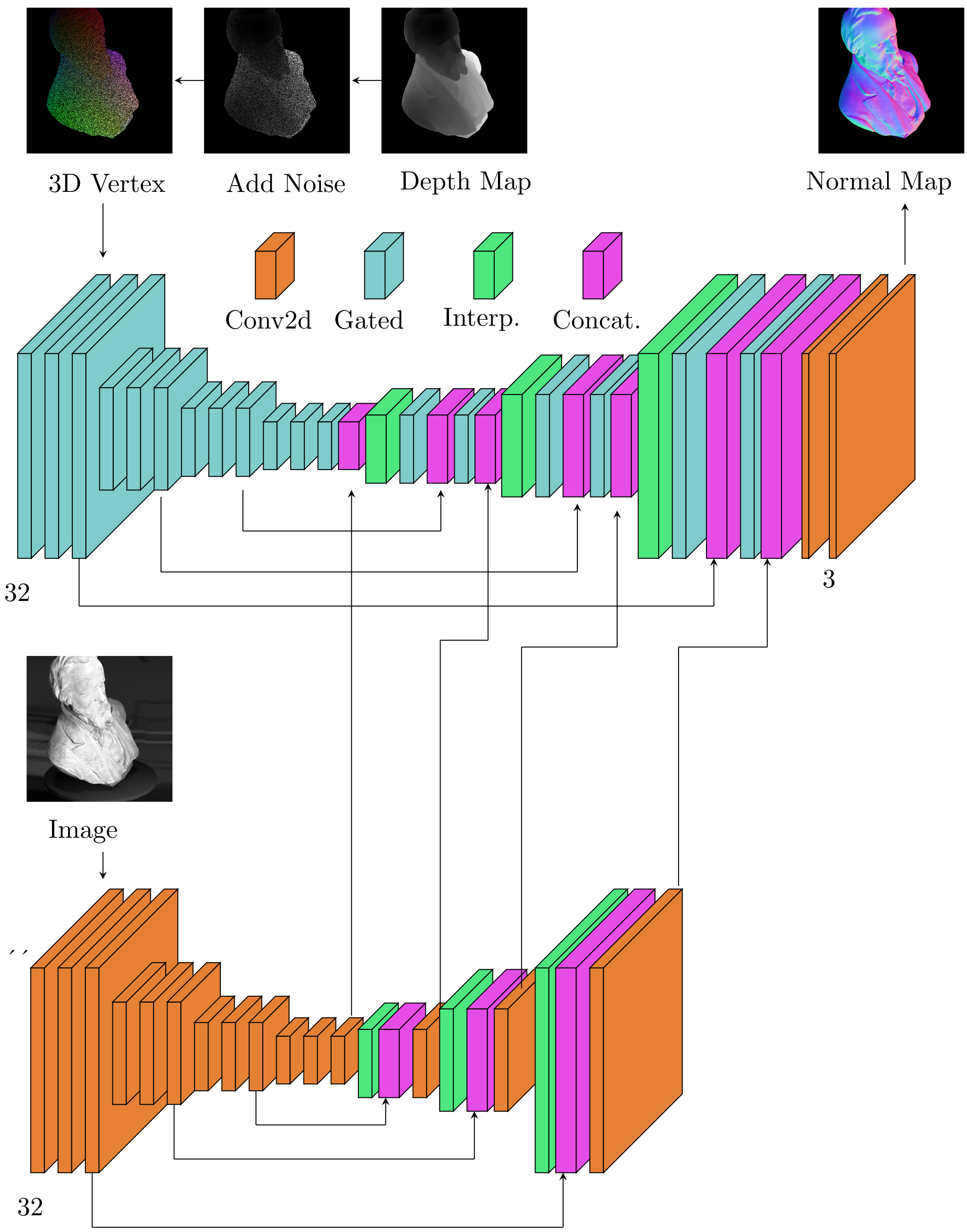


FIGURE 3.3: Guided Gated Convolution Neural Network for normal estimation. The normal branch shows on the upper side taking point cloud as input. The image branch shows on the lower side taking image as input. There are total 4 times fusions between the two branches. The output is a corresponding normal map.

Chapter 4

Dataset

4.1 Synthetic Dataset

To train a deep learning model with supervised learning scheme, a dataset should require two principles, truth-worthy ground-truth and comprehensive scenarios. The depth map captured by Kinect is not satisfied the first requirement since it is usually semi-dense with a number of missing pixels, as shown in Figure 1.2. Therefore, a more elaborate depth map is required for the training work. In this thesis, a dataset called "synthetic50-5" is created for the training works.

4.1.1 Resource

The Stanford 3D Scanning Repository n.d., McGuire, 2017, McGuire, n.d. and *Smithsonian 3D Digitization* n.d. published a set of point cloud dataset on the internet for computer vision task research. These point clouds are scanned from real objects using high resolution scanners like Cyberware 3030 MS+ and calibrated with post processing. Each objects has been scanned for hundreds of times for an exhaustive completion for the origin objects, which is up to millions points. (*The Stanford 3D Scanning Repository* n.d.). The dense point clouds makes the normal inference task trivial since the neighbor based method performs good enough for these kind of task. Some of the point cloud even equipped with pre-computed normal map based on more advanced methods. They all provides the accurate ground-truth for the supervised learning method.

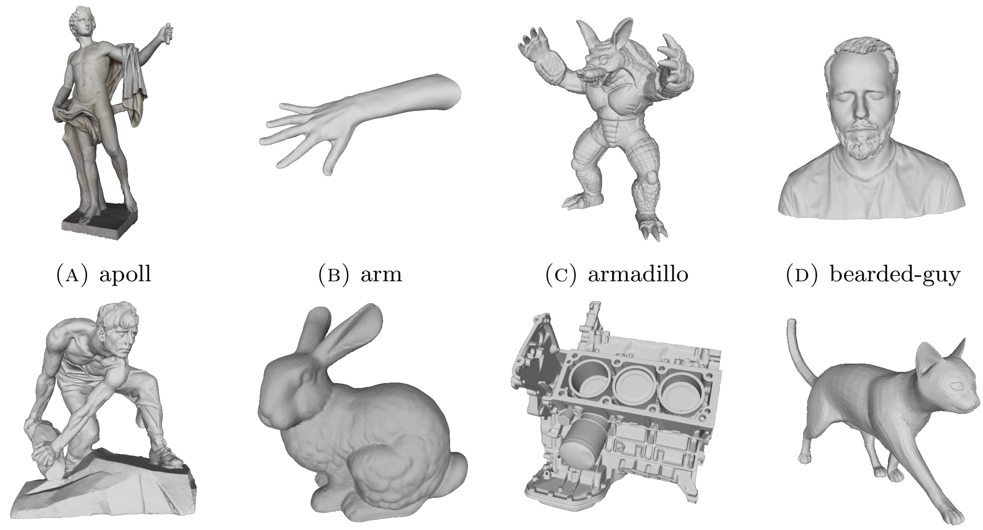
The "synthetic50-5", is a datset with 50 point clouds as training set and 5 point clouds as test set. The dataset is created base on the work of this thesis and using for normal inference task. Figure **??** gives the illustrations of some objects. Appendix A gives a full version of dataset models.

4.1.2 Synthesize Scenes using Unity

In order to fit the using scenario of Kinect as much as possible, the dataset consists of the generated synthetic 3D scenes via Unity, which is a game engine using for 3D games creation.

In the synthetic scenes from engine, the object is placed on a cylinder platform, which is lighted by a directional light nearby. A camera focus on the platform and captured the scene. The layout in the game engine is shown in Figure 4.2. In order to simulate more scenarios, the positions of the camera and directional light are randomly changed in each new scene. For 50 training objects, 1000 scenes are generated and each scene is saved in 3 kinds of resolutions $512 \times 512 \times 3$, $256 \times 256 \times 3$ and $128 \times 128 \times 3$.

The main advantage using generated scene is the availability of complete information. The depth map can be captured in a loss-free way. The corresponding normal



(A) apoll (B) arm (C) armadillo (D) bearded-guy

(E) bronze-sculpture (F) bunny (G) car-engine (H) cat

FIGURE 4.1: Point clouds scanned by high resolution scanners

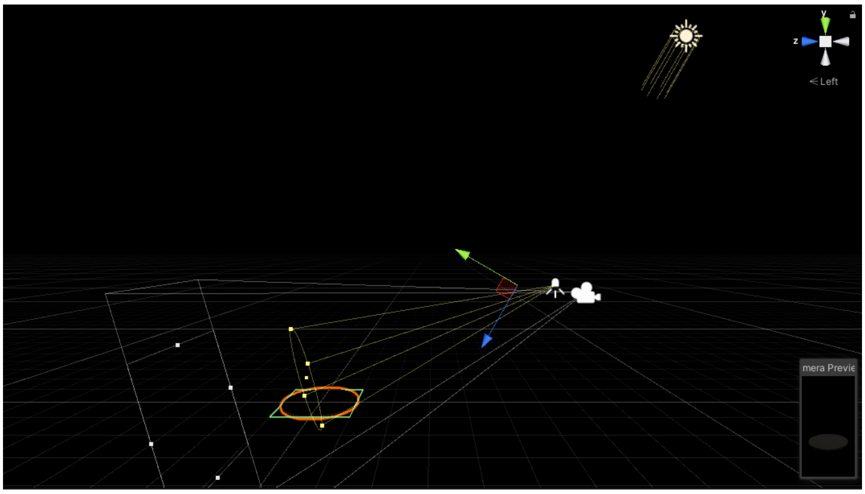


FIGURE 4.2: The layout of synthetic scenes generation in Unity.

map can also be safely considered as ground truth. And the scale of the dataset is easy to control.

For each scene, following information is recorded

A depth map D is captured by a depth camera in Unity, which is a 1 channel image that contains the information relating to the distance of the surfaces of the scene objects from a viewpoint. It can be saved as a 16-bit grayscale image, i.e. each pixel in range $0-65535$. The grayscale image can be used as guided normal inference task and also as a readable scene for human. The gray-color is converted from RGB color based on following equation

$$gray : \frac{r + 2g + b}{4}$$

The normal map is the tangent surface normal, which is saved in 32-bit RGB color image. The surface normal (n_x, n_y, n_z) and its corresponding RGB color (R, G, B)

TABLE 4.1: The information saved for each scene in "synthetic50-5".

Data	**Size**
Depth map	$512 \times 512 \times 3$
Depth range	2×1
Grayscale Image	$512 \times 512 \times 1$
Normal Map	$512 \times 512 \times 3$
Light Position	3×1
Camera Intrinsic Matrix	3×3
Camera Extrinsic Matrix	3×4

can be converted based on following equation:

$$
\begin{aligned}
n_x &= \frac{R}{255} * 2 - 1 \\
n_y &= \frac{G}{255} * 2 - 1 \\
n_z &= 1 - \frac{B}{255} * 2
\end{aligned}
$$

If consider the relation between Lambertian reflection and normal direction, the light source can be used to calculate the reflect direction of each point. The camera intrinsic and extrinsic matrix helps point cloud calculation.

It is necessary to point out again that "synthetic50-5" aiming to rebuild the using scenarios of Kinect, where all types of the generated data files shown in Table 4.1 are also the same format of the Kinect data.

4.1.3 Convert to Point Cloud

The depth map can be converted to 3D vertex point cloud as the input of the normal inference model.

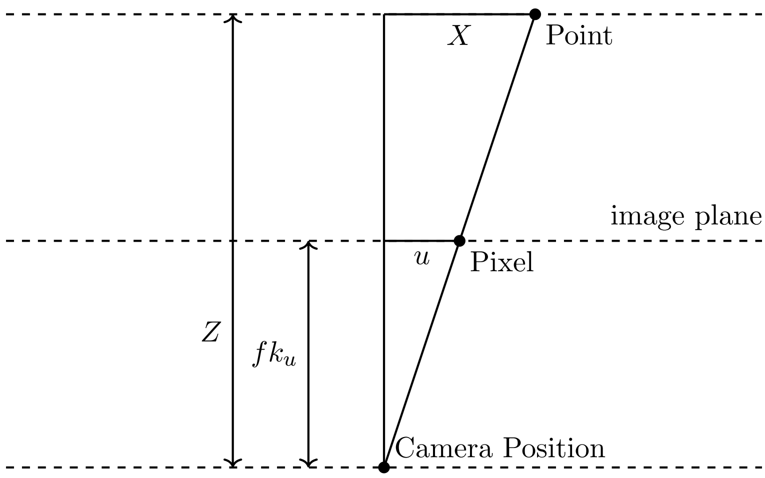


FIGURE 4.3: Convert depth to point in camera coordinate system

Consider a 3-dimensional Euclidean space. Use z axis denotes the depth. The x and y axes perpendicular with each other. For a pixel (u, v) on depth map, its depth

$D(u, v)$ is the Z component of the corresponding point $P_C = (X, Y, Z)$ in camera coordinate system. The X and Y can be calculated based on the triangle similarity

$$X = \frac{uZ}{fk_u}$$
$$Y = \frac{vZ}{fk_v}$$

where fk_u, fk_v is the focal length in pixels align u and v axes. Converted a point from camera coordinate system to world coordinate system, using extrinsic matrix R and t

$$P_W = P_C R + t$$

4.1.4 Point Cloud Normalization

The sizes of each training object are various, whereas it should be as an invariant value for the training model. Thus the normalization is required before feed training objects into the models. The range of each axis is shown in Figure **??**. Table 4.2 gives a quantitative measurement of corresponding average values.

The normalization has been performed as follows. First translate the points to the original point as close as possible, then choose the range value of one axis as a scale factor, normalize the points to unit vectors. The equation is shown as follows

$$X_n = \frac{X - \min(X)}{s}$$
$$Y_n = \frac{Y - \min(Y)}{s}$$
$$Z_n = \frac{Z - \min(Z)}{s}$$

where s is a scale factor,

$$s = \max(X) - \min(X)$$

It is calculated as the range in X axis, but theoretically can be used by Y or Z axes as well.

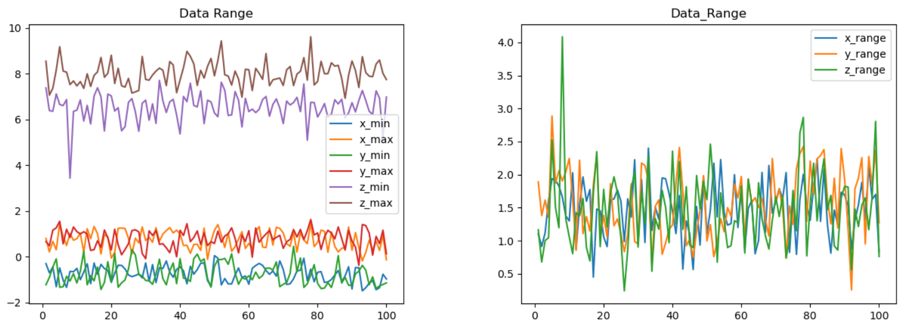


FIGURE 4.4: Left: Extreme value in 3 axis; Right: Vertex range in 3 axis

TABLE 4.2: The fluctuation of extreme values and their ranges in 100 random training items.

Axis	**Scale**	**Min**	**Max**
X	1.48	-0.75	0.73
Y	1.56	-0.76	0.80
Z	1.47	6.53	8.00

TABLE 4.3: The structure of a single tensor in the dataset.

Name	**Content**
input-tensor	Vertex Image Light Direction
output-tensor	GT-Normal Image GT-Light-Direction
Light position	light position
Camera Matrix	K,R,t
Depth Range	minDepth, maxDepth

4.1.5 Noise

The raw depth maps captured by Kinect usually have missing pixels. Therefore, the "synthetic50-5" dataset adds a similar properties. As shown in Figure **??**. the depth map has missing pixels distributed all around the scene. Correspondingly, an uniformly distributed pixel-delete noise is used for noise simulation. Furthermore, a parameter μ is used to control the intensity of noise, it denotes the μ-percent pixel dropoff. For example, $\mu - 10$ means removes 10% pixels randomly. For each scene, the noising operation based on a random μ in a range $[0, 50]$, therefore some scenes have more missing pixels and some have less. The random noise intensity also enables the model to learn scenarios not only with noise, but also with minor noise or even without noise. Figure 4.5 shows the noise effect on different μ.

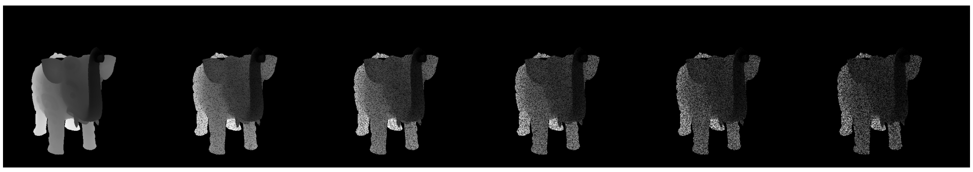


FIGURE 4.5: Noise-intensity on $\mu{-}0, \mu{-}10, \mu{-}20, \mu{-}30, \mu{-}40, \mu{-}50,$. Object Name: elephant-zun-lid.

4.1.6 Fit to PyTorch

In order to saving the training time, the dataset is compressed in PyTorch format. The structure of a single item is shown in Table 4.3.

4.2 Real Dataset

The real dataset is the depth map and rgb image captured via Kinect...

Chapter 5

Experiments

The experiments are performed on two normal inference tasks: normal inference based on depth image and guided normal inference based on RGB-D image. Prior works for normal estimation using very deep networks. For a single object surface normal detection as stated in this thesis, the given methods has a similar performance but only with 1/10 size.

The model is trained with PyTorch 1.10.2, CUDA 10.2.89, GPU with single NVIDIA GEFORCE GTX 1080Ti.

5.1 Surface Normal Inference based on Depth Map

The goal of surface normal inference is to calculate the tangent surface normal map N from a single depth map D. A network named Gated Convolution Neural Network (GCNN) is trained and is compared with similar approaches.

5.1.1 Compared Approaches

5.1.2 Training Details

The network (see 3.2) is trained on stated dataset in Chapter 4. It has two versions, the L-size $512 \times 512 \times 3$ and the S-size $128 \times 128 \times 3$. For the L-size model, it uses batch size 8 for 3500 iterations. For the S-size model, it uses batch size 32 for 6000 iterations. Both sizes use Adam optimizer Kingma and Ba, 2014 with a learning rate of 1×10^{-3}, penalty-l2 loss(see **??**). The output is directly the tangent normal in range $[-1, 1]$. The output and input has the same shape.

5.1.3 Qualitative Evaluation

The evaluation visualization on dataset "synthetic50-5" (see 4) dataset is shown in Figure 5.1. It compares the results with method xxx, for both smooth and highly detailed cases. In all the cases, the angle errors are very close. The test-cases are based on test point cloud in "synthetic50-5".

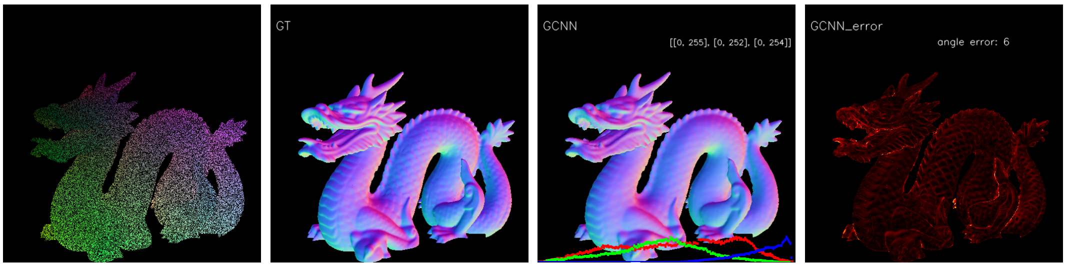
The model is supposed to used on any image size since it has no fully-connected layers (dense layers). Two size of models are based on exactly the same architecture.

It is

The evaluation visualization on real dataset is shown in Figure 5.2

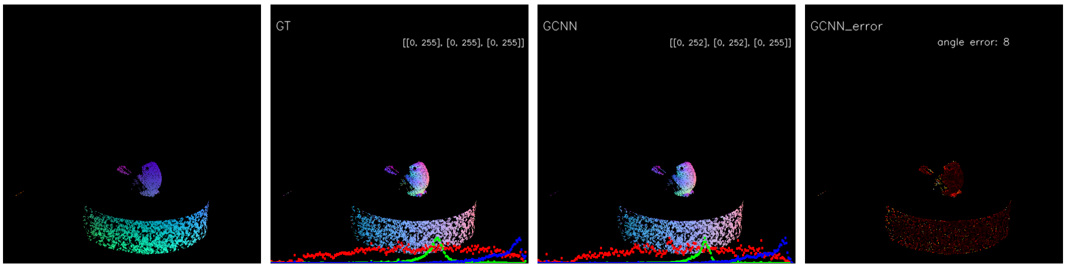
5.1.4 Quantitative Evaluation

The evaluation result on test scenes is shown in Figure 5.3. The GCNN based method has angle error between 5 to 15 degrees in both type of inputs. The error trends to higher with point number decrease. It is because the less points in the point cloud,



(A) point cloud (B) ground-truth (C) GCNN (D) Angle Error

FIGURE 5.1: GCNN Normal Inference on Synthetic Dataset (object: dragon)



(A) point cloud (B) ground-truth (C) GCNN (D) Angle Error

FIGURE 5.2: Evaluation on Real Dataset

the more detail is hided due to the insufficient of resolution. Therefore the recorded surface based on the point cloud is more coarse, which also increase the difficulty of the normal inference.

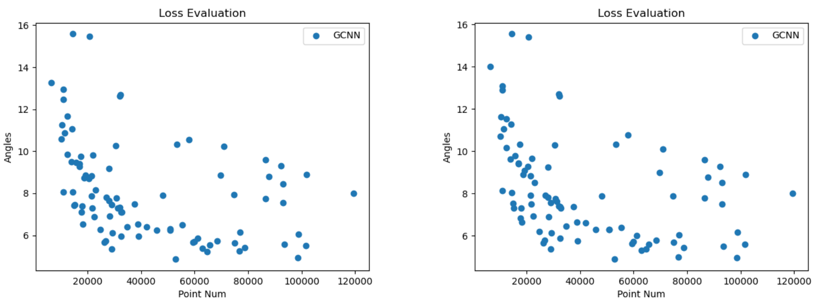


FIGURE 5.3: Evaluation of average angular loss on the whole test dataset with 90 scenes. The x-axis indicates the point number, the y-axis indicates the angles. The **Left** one using point cloud without noise, the **right** one has noise.

5.1.5 Speed

5.2 Guided Gated Convolution Neural Network for Normal Inference

The inference result of guided-GCNN model is shown in Figure 5.4. With adding the information of a gray-scale image, the model is able to sharpen the details over the whole scene.

(A) GT (B) g-GCNN (C) Angle Error (D) GCNN (E) Angle Error

FIGURE 5.4: guided-GCNN Normal Inference on Synthetic Dataset (object: dragon). GCNN result is shown on (d) and (e) as comparison.

5.3 Guided Gated Convolution Neural Network for Normal Inference

From the Figure 5.5 we can observe the normal difference between ground-truth and GCNN predicted normals in another dimension. It separates the interval $[-1, 1]$, which is exactly the range of normal vector, to 256 sections. Then it counts the number of points locates in each section for 3 axes. The 3 axes are fitted quit well in most of interval but other than $[-0.25, 0.25]$ for x and y axes and interval close to -1 for z axis. Therefore a further constraint can be considered to the loss function related to the normal difference shown in this figure.

It is faulty that almost no normal has -1 z-component in GCNN predicted normal map. The reason?

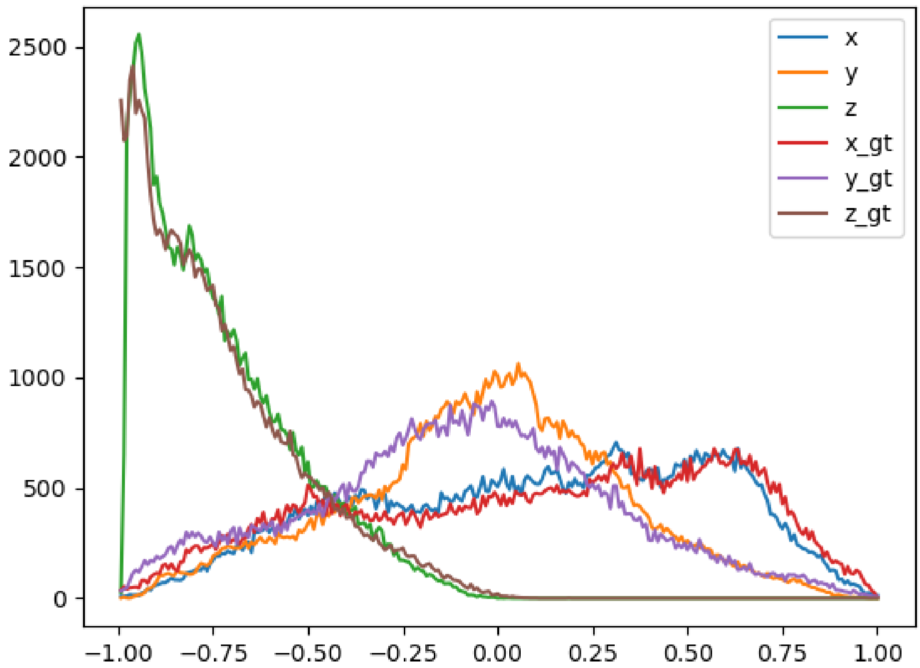
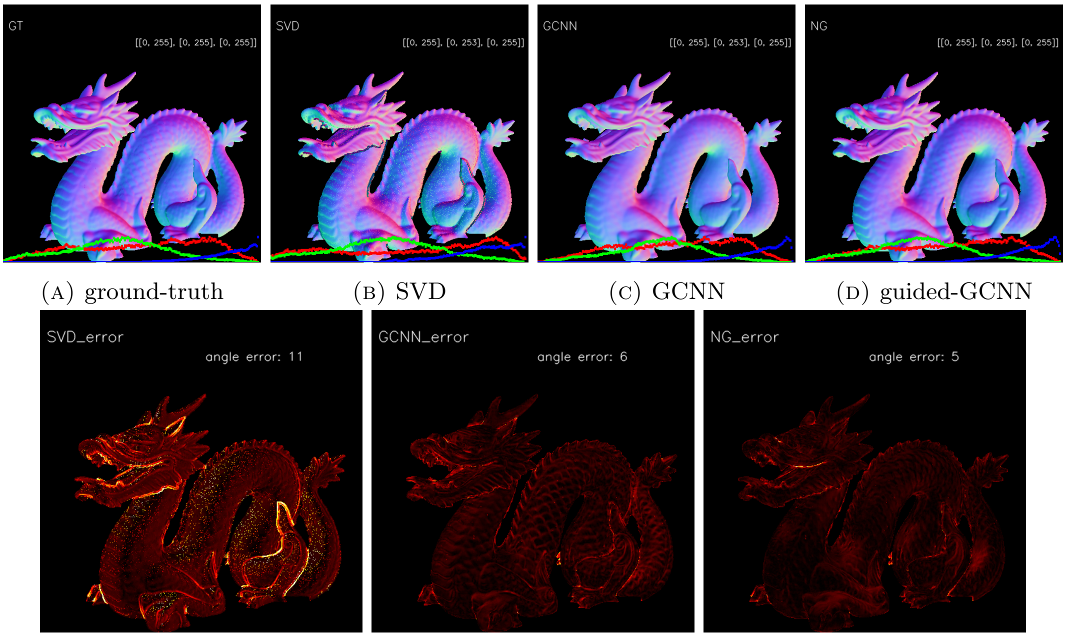


FIGURE 5.5: The normal difference of between GCNN and ground-truth in x, y, z-axis respectively. The y axis indicates the number of points, x axis indicates the value of normal in x/y/z axis. (The chart is based on the "dragon" scene showing above)

5.4 model comparison

This section evaluates the proposed models with the neighbor-based model and make comparison with each other.



(A) ground-truth (B) SVD (C) GCNN (D) guided-GCNN

(E) Error (F) Error (G) Error

FIGURE 5.6: Normal Inference on different models with errors. (object: dragon)

Chapter 6

Conclusion

Gated convolution neural network...

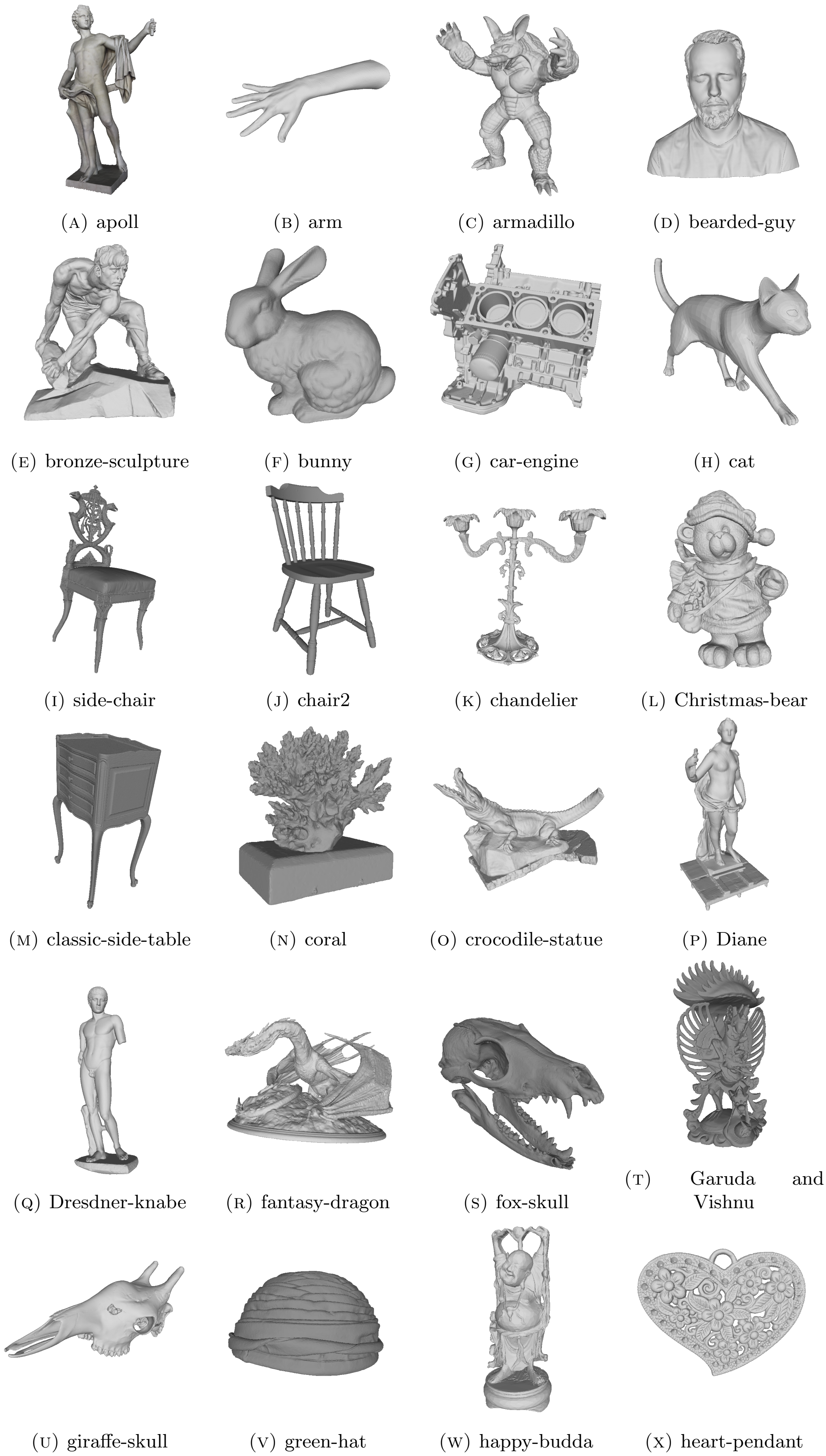
Appendix A

Dataset

A.1 Dataset

A.2 How do I change the colors of links?

The color of links can be changed to your liking using:
`\hypersetup{urlcolor=red}`, or
`\hypersetup{citecolor=green}`, or
`\hypersetup{allcolor=blue}`.
If you want to completely hide the links, you can use:
`\hypersetup{allcolors=.}`, or even better:
`\hypersetup{hidelinks}`.
If you want to have obvious links in the PDF but not the printed text, use:
`\hypersetup{colorlinks=false}`.



(A) apoll (B) arm (C) armadillo (D) bearded-guy

(E) bronze-sculpture (F) bunny (G) car-engine (H) cat

(I) side-chair (J) chair2 (K) chandelier (L) Christmas-bear

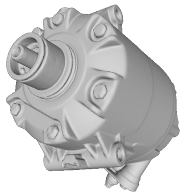
(M) classic-side-table (N) coral (O) crocodile-statue (P) Diane

(Q) Dresdner-knabe (R) fantasy-dragon (S) fox-skull (T) Garuda and Vishnu

(U) giraffe-skull (V) green-hat (W) happy-budda (X) heart-pendant

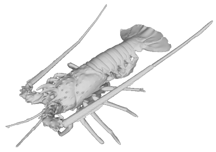
FIGURE A.1: Point clouds in training dataset A

(a) indonesian-statue

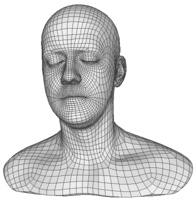


(b) industrial-compressor

(c) kylix



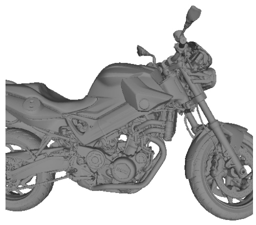
(d) lobster



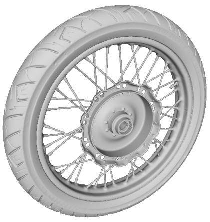
(e) head

(f) Lu-Yu

(g) Maitreya

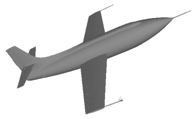


(h) motorbike



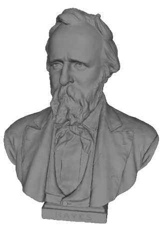
(i) motorcycle-wheel

(j) Pan



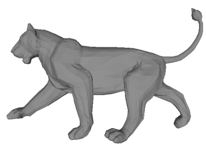
(k) plane

(l) elkins-refrigerator



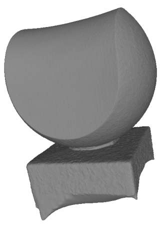
(m) Rutherford

(n) statue-dragonfly-tamer



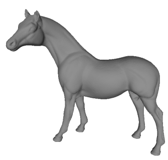
(o) Tiger

(p) transmission



(q) universe

(r) Vestalin

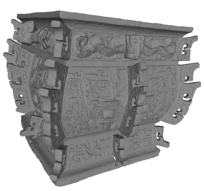


(s) Zebra

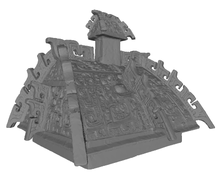
(t) elephant-zun-base

(u) elephant-zun-lid

(v) cosmic-budda



(w) fangyi-base



(x) fangyi-lid

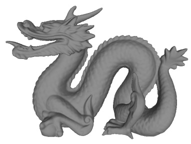
Figure A.2: Point clouds in training dataset B

(A) guang-base

(B) guang-lid

(C) Garfield

(D) bus



(E) dragon

(F) baoshanlu

(G) Washington

FIGURE A.3: Point clouds in training dataset C

Bibliography

Ben-Shabat, Yizhak, Michael Lindenbaum, and Anath Fischer (2019). "Nesti-Net: Normal Estimation for Unstructured 3D Point Clouds Using Convolutional Neural Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cho, Kyunghyun et al. (2014). *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. DOI: 10.48550/ARXIV.1409.1259. URL: https://arxiv.org/abs/1409.1259.

Eigen, David, Christian Puhrsch, and Rob Fergus (2014). *Depth Map Prediction from a Single Image using a Multi-Scale Deep Network*. DOI: 10.48550/ARXIV.1406.2283. URL: https://arxiv.org/abs/1406.2283.

Eldesokey, Abdelrahman, Michael Felsberg, and Fahad Shahbaz Khan (2020a). "Confidence Propagation through CNNs for Guided Sparse Depth Regression". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.10, pp. 2423–2436. DOI: 10.1109/tpami.2019.2929170. URL: https://doi.org/10.1109%2Ftpami.2019.2929170.

— (2020b). "Confidence Propagation through CNNs for Guided Sparse Depth Regression". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.10, pp. 2423–2436. DOI: 10.1109/tpami.2019.2929170. URL: https://doi.org/10.1109%2Ftpami.2019.2929170.

Eldesokey, Abdelrahman et al. (2020). *Uncertainty-Aware CNNs for Depth Completion: Uncertainty from Beginning to End*. DOI: 10.48550/ARXIV.2006.03349. URL: https://arxiv.org/abs/2006.03349.

Fouhey, David F., Abhinav Gupta, and Martial Hebert (2013). "Data-Driven 3D Primitives for Single Image Understanding". In: *2013 IEEE International Conference on Computer Vision*, pp. 3392–3399. DOI: 10.1109/ICCV.2013.421.

He, Kaiming et al. (2015). *Deep Residual Learning for Image Recognition*. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

Hochreiter, Sepp and Jürgen Schmidhuber (Dec. 1997). "Long Short-term Memory". In: *Neural computation* 9, pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

Holzer, S. et al. (2012). "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2684–2689. DOI: 10.1109/IROS.2012.6385999.

Horn, Berthold (Oct. 2004). "Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View". In: URL: http://hdl.handle.net/1721.1/6885.

Hua, Jiashen and Xiaojin Gong (2018). "A Normalized Convolutional Neural Network for Guided Sparse Depth Upsampling". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. IJCAI'18. Stockholm, Sweden: AAAI Press, 2283–2290. ISBN: 9780999241127.

Johnson, Justin, Alexandre Alahi, and Li Fei-Fei (2016). *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*. DOI: 10.48550/ARXIV.1603.08155. URL: https://arxiv.org/abs/1603.08155.

Kingma, Diederik P. and Jimmy Ba (2014). *Adam: A Method for Stochastic Optimization*. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

Knutsson, Hans and Carl-Fredrik Westin (Jan. 1993). "Normalized and Differential Convolution Methods for Interpolation and Filtering of Incomplete and Uncertain Data". In.

Laina, Iro et al. (2016). *Deeper Depth Prediction with Fully Convolutional Residual Networks*. DOI: 10.48550/ARXIV.1606.00373. URL: https://arxiv.org/abs/1606.00373.

McGuire, Morgan (n.d.). *Artec3D*. URL: https://www.artec3d.com/3d-models/art-and-design.

— (2017). *Computer Graphics Archive*. URL: https://casual-effects.com/data.

NETGEN An advancing front 2D/3D-mesh generator based on abstract rules (1997). DOI: 10.1007/S007910050004.

Oord, Aaron van den et al. (2016). *Conditional Image Generation with PixelCNN Decoders*. DOI: 10.48550/ARXIV.1606.05328. URL: https://arxiv.org/abs/1606.05328.

Qi, Fei et al. (2013). "Structure guided fusion for depth map inpainting". In: *Pattern Recognition Letters* 34.1. Extracting Semantics from Multi-Spectrum Video, pp. 70–76. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2012.06.003. URL: https://www.sciencedirect.com/science/article/pii/S0167865512001912.

Qi, Xiaojuan et al. (2018). "GeoNet: Geometric Neural Network for Joint Depth and Surface Normal Estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Redmon, Joseph and Ali Farhadi (2018). *YOLOv3: An Incremental Improvement*. DOI: 10.48550/ARXIV.1804.02767. URL: https://arxiv.org/abs/1804.02767.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. arXiv: 1505.04597 [cs.CV].

Smithsonian 3D Digitization (n.d.). URL: https://www.3d.si.edu/explore.

Tan, Mingxing, Ruoming Pang, and Quoc V. Le (2019). "EfficientDet: Scalable and Efficient Object Detection". In: DOI: 10.48550/ARXIV.1911.09070. URL: https://arxiv.org/abs/1911.09070.

The Stanford 3D Scanning Repository (n.d.). URL: http://www.graphics.stanford.edu/data/3Dscanrep/.

Yang, Na-Eun, Yong-Gon Kim, and Rae-Hong Park (2012). "Depth hole filling using the depth distribution of neighboring regions of depth holes in the Kinect sensor". In: *2012 IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC 2012)*, pp. 658–661. DOI: 10.1109/ICSPCC.2012.6335696.

Yu, Jiahui et al. (2018). *Free-Form Image Inpainting with Gated Convolution*. DOI: 10.48550/ARXIV.1806.03589. URL: https://arxiv.org/abs/1806.03589.

Zhou, Jun et al. (2021). *Fast and Accurate Normal Estimation for Point Cloud via Patch Stitching*. arXiv: 2103.16066 [cs.CV].