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

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

July 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

I	Grayscale Image Matrix (Height $\times$ Width $\times$ 1)
L	Light Map Matrix (Height $\times$ Width $\times$ 3)
N	Normal Map Matrix (Height $\times$ Width $\times$ 3)
V	Vertex Map Matrix (Height $\times$ Width $\times$ 3)
X	Input Matrix (Height $\times$ Width $\times$ Channels)
Y	Output Matrix (Height $\times$ Width $\times$ Channels)
w	Width of Matrix
h	Height of Matrix

List of Symbols

$[AB]$	concatenation between A and B
$\oplus$	element-wise multiplication
$\cdot$	dot product

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. However, the calculation of surface normal in many tasks is not as straightforward as simply the cross-product of two plane vectors. 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 to save the fine detail of the objects and also can be easier measured by 3D scanners. Therefore the task is converted to the normal inference based on the surface point, and the most of the case, the result surface normal is merely an approximation.

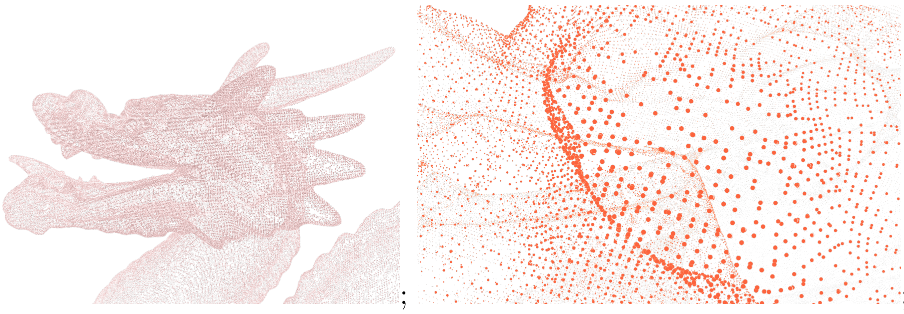


FIGURE 1.1: Left: A part of the point cloud of the dragon model. Right: The zoom in of the left point cloud.

Apart from this, due to the application scenarios, the working principle of scanners are various, which consequently produce point cloud structures in different forms. 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. It increases 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.

However, for the scanner with calibrated camera, a structured point cloud can be captured. One example is the depth camera. It captures the RGB-D images for the object, which includes the standard RGB image with depth information of each pixel. After camera calibration, a corresponding point cloud can be calculated based on the depth map and camera matrix. It gives the advantage that 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. It provides a better view for the normal inference task since the neighbor information can be considered as an important reference. Besides the point cloud, which is one piece of information can be used for normal reference, the

RGB image also leaves the trace of surface normal. If assume object surface is diffuse and the light coming from a knowing direction , the brightness of the object surface is proportional to the surface normal, which is known as Lambertian surface. It indicates the normal inference task can also utilize the relationship between surface normal and light reflection to further rectify the inferred normal map with knowing light source. Therefore, the illuminated calibrated RGB-D image provides as a good input for normal inference tasks. Unfortunately, in the actual situation, the depth maps captured by the sensors are only semi-dense, which is mainly caused by optical noises and the reflections in dark and shinny areas. Consequently, it disrupts the robustness of the normal inference methods. Median filters can be used for the sparse missing pixels, 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.

Deep learning based methods provides multiple possible solutions for the challenges mentioned above. First, to deal with the noised input, deep learning based methods already have the solution for the similar tasks like image inpainting and depth density enhancement, which base on the noised image as input to predict the clean and fully dense output with or without original meanings. Therefore, it can be a help for noise in the depth map. Besides, the network of deep learning model can also handle the structured point cloud as a single input and infer the corresponding normal all together, which is very time economical comparing to the approaches like neighbor based methods. In addition, the multi-stages training architectures provide a way to consider both depth map and texture image as the input to predict the surface normals, which not only consider the point cloud but also the add the lambertian reflection as a further constraint.

In this work, we focus on the normal inference based on a semi-dense depth map with RGB image using deep learning based approaches. Specifically, the missing pixels in the depth map is filled up by the gated convolution and propagate in a customized UNet with skipping connections. The output of the training model is directly the surface normal corresponding the input depth map. The grayscale image is used to further improve the estimation accuracy. For the training work, a dataset named "synthetic50-5" is created including 55 high resolution point clouds from internet, as shown in Appendix A. The point clouds provide with the high accurate normals, which can be used as the ground truth of the training work. Most of the models have elaborate details with high curvatures but also contain smooth surfaces. The trained model is evaluated on both synthetic dataset as well as the real dataset captured from RGB-D cameras. A series of metrics have been used for qualitative evaluation. The model is shown to achieve a remarkably better prediction accuracy at a low computational cost compared to the standard approaches for semi-dense point clouds.

The structure of the thesis is as follows, Chapter 1 is the introduction of the whole work. Chapter 2 briefly discusses the related work about normal inference. Chapter 3 is the main approaches of this work. Chapter 4 introduces the created dataset for the training work. Chapter 5 is the description of the experiments and the evaluation of the models. Chapter 6 is the conclusion of the whole thesis.

Chapter 2

Related Work

Due to the different of data structures, the point clouds can be grouped into types, structured point cloud based and unstructured point cloud based. The first case contains the neighbor information of each point together with a depth map or RGB image, the second case does not contain any neighbor information which has to be further calculated.

Structured Point Cloud Based The relevant methods derive the surface normals based on spatial relationship, which utilizes the neighbor information for estimation. These methods 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 earlier methods usually use optimized methods. Holzer et al., 2012 proposed method to use local neighbors with an interest parameter p and compute the eigenvectors of the corresponding covariance matrix. They also 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. Klasing et al., 2009 did a comparison among the optimization-based methods.

Unstructured Point Cloud Based For the unstructured point cloud, the neighbor information of each point is usually unknown. K-nearest neighbor (KNN) is a common algorithm for neighbor searching. With knowing this information, the neighbor based approaches can be used as a second step. However, KNN-method merely based on the Euclidean distance in the 3D space. Therefore, the points of the other surfaces but in a close distance will also be considered as neighbors. To ease this problem, Ben-Shabat, Lindenbaum, and Fischer, 2019 proposed a method based on unstructured point cloud with selecting the optimal scale around each point for normal estimation. To calculate the normals of multiple points in parallel Zhou et al., 2021 processes a series of overlapping patches for normal estimation.

Deep learning based methods take single RGB image or RGB-D image as the input for normal estimation. It has a strong relationship with the depth inference tasks, two benchmarks are highly used in these area. (Silberman et al., 2012 and Uhrig et al., 2017) Based on the input of training model, the methods can be roughly divided as follows:

Depth Based Depth map contains the spatial information of the object surface, which is very important for the normal inference task. However, the depth map captured by depth sensors are usually with missing pixels and holes on dark, shinny or transparent regions.(Silberman et al., 2012). To overcome the missing pixels, Yang, Kim, and Park, 2012 proposed a depth hole filling method using the depth distributions of neighboring regions. Knutsson and Westin, 1993 introduced normalized convolution dealing with missing or uncertain data for convolution operation. It uses a binary mask to distinguish missing data and integrate it into the convolution operation. Eldesokey, Felsberg, and Khan, 2020a applied it and use normalized convolution layers in their networks, which aims to reconstruct the missing pixels from the sparse depth map sensed by cameras. Eldesokey et al., 2020 proposed an input confidence estimation network to estimate the confidence instead of using a binary mask. However, the relevant papers are only using 1 channel data as model input, which didn't discussed the case for the multiple-channel data as input, such as RGB image, or structured point cloud. Hua and Gong, 2018 further integrated RGB image as the guidance to deal with the missing pixels in the depth map.

RGB based RGB based methods predict the depth map diretly from single RGB image. Eigen, Puhrsch, and Fergus, 2014 proposed a two staged network for depth map prediction based on RGB image, which consider the global features and the local features respectively. Laina et al., 2016 employed Residual Network for the feature extraction and further designed a upsampling part which replace the fully-connected layer with the unpolling layers. Fouhey, Gupta, and Hebert, 2013 proposed a method to learn discriminative and geometrically informative primitives from RGBD images, which is further used to recover the surface normals of a scene from a single image. Qi et al., 2018 uses ResNet (He et al., 2015) to infer a coarse surface normal based on RGB image, and further refine it with the help of depth map based on the methods based on Fouhey, Gupta, and Hebert, 2013. Li et al., 2022 proposed a method achieved state of art in NYUv2 Dataset (Silberman et al., 2012), which adaptively generate information to predict depth maps based on RGB image.

RGB-D based Zeng et al., 2019 based on UNet architecture for normal estimation using RGB-D image as input. The RGB image and depth map are in the separate branches and imply a fusion module in four sections of the network to concatenate two branches. It also considers the confidence of the values in depth map.

Chapter 3

Approaches

The normal inference task estimate the surface normal of a 3D object. There are two main approach to solve the task. Geometry based approach estimate the surface normal of an object based on the geometry principle. Point cloud is a common surface geometry information which samples the surface point position in 3D space. Another approach is photometric stereo based approach, which utilizes a set of images under different illumination conditions. In this chapter, geometry and photometric stereo based approach are introduced separately, then a deep learning based approach using geometry information is proposed. In the end, as the main work of the thesis, another deep learning method is proposed based on both geometry and photometric stereo information.

3.1 Geometry based normal estimation

3.1.1 Approach

The geometry based normal estimation uses point cloud of the object surface as input. Given a structured point cloud $V^{W\times H\times 3}$ to estimate the normal map $N^{W\times H\times 3}$, where each normal $\mathbf{n}^{3\times 1}$ at point $\mathbf{v}^{3\times 1}\in N$ is a unit vector with its direction point outward of the surface and perpendicular to the tangent plane of the surface at point $\mathbf{p}^{3\times 1}$.

The idea behind the neighbor based method is to fit a plane Π using the k neighbors $\mathbf{p}_1, ..., \mathbf{p}_k \in \mathbb{R}^3$ of the point $\mathbf{p}$, calculate the normal $\tilde{\mathbf{n}}$ of the plane.

It is under the assumption that the point and its neighbors are located in the same plane. This is usually not hold for the most of the surface, but if k in a suitable scale and point cloud is dense enough, it is enough to get an accurate and sharp result. As shown in Figure **??**.

Specifically, it is not necessary to find the exact plane equation to solve the normal. Instead, the normal can be derived based on an equation system. The normal $\tilde{\mathbf{n}}$ of plane Π is perpendicular to all the vector on the plane Π, we can construct k vectors on plane Π use k neighbors $\mathbf{p}_1, ..., \mathbf{p}_k \in \mathbb{R}^3$ of point $\mathbf{p}$. For the simplicity, we can choose $\mathbf{p}_1$ as the base point, then $k-1$ vectors can be construct as follows

$$\mathbf{v}_i = \mathbf{p}_1 - \mathbf{p}_i \quad \text{for } i = 2, ..., k \tag{3.1}$$

and each of them satisfied $\mathbf{v}_i \cdot \tilde{\mathbf{n}} = 0$. Then, the equation system can be constructed as

$$A \cdot \mathbf{n} = 0 \tag{3.2}$$

where $A \in \mathbb{R}^{(k-1)\times 3}$ is the vector matrix vertically stacked by $\mathbf{v}_1, ..., \mathbf{v}_{k-1}$. 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.

To calculate a valid normal, at least 3 points are required to construct, i.e. $k >= 3$. For the sake of robust, more points can be used to reduce the measuring error. For the case $k > 3$, since the surface vectors are actually not in the same plane, the equation system is likely over-determined . Then the equation system mentioned above can be converted to follow optimization problem

$$\begin{aligned} \min \quad & \|A\mathbf{n}\|^2 \\ \text{s.t.} \quad & \|\mathbf{n}\|^2 = 1 \end{aligned} \tag{3.3}$$

which can be solved by singular value decomposition(SVD). Let the decomposition of

$$A = U\Sigma V^T$$

The solution i.e. normal is the last column of V.

At last, all the normals should point ot view point $\mathbf{s}$, thus the direction of a normal should be inverted if

$$\mathbf{n} \cdot (\mathbf{p} - \mathbf{s}) > 0 \tag{3.4}$$

Repeat the procedure for all the points in the point cloud to get the entire normal map.

3.1.2 Evaluation

The neighbor based method can predict the normal map in a good way when the given point cloud is dense, as shown in Figure **??**. It can successfully predict the smooth surface of the dragon object, especially the flakes and the tails of the dragon.

However, it failed in the areas such as hindleg, horn and the mouth, which consists mainly by sharp edges. This is because the neighbors points in these area do not hold the assumption of coplanarity well, the normals of these neighbors can be very different. The neighbor based method is depended on a well-chosen parameter k.

FIGURE 3.1: Normal map of a dragon object predicted by neighbor based method. k=2, angle error=5 **Left**: ground-truth normal map **Middle**: predicted normal map, **Right**: Error map

Figure **??** shows the evaluation on different k values. When $k = 1$, the average error of the whole image is the lowest one, most of the normals are close to the ground-truth but the outline edges, which are the areas that surface normal changed extremely sever. For the case $k = 2$, the sharp edges are more smooth and cause more error, like the eyes area of the dragon. Compare to the first case, the outline edge error goes better. Most of the edge errors are reduced when $k = 2$, since more neighbor points join the evaluation and it reduces the effect of outliers. However, for the area of horn outline, hindleg outline, the error goes worse. In this case, most of the neighbors of these points are outliers and thus failed this approach. $k = 3$ and $k = 4$ further increase the angle errors based on $k = 2$.

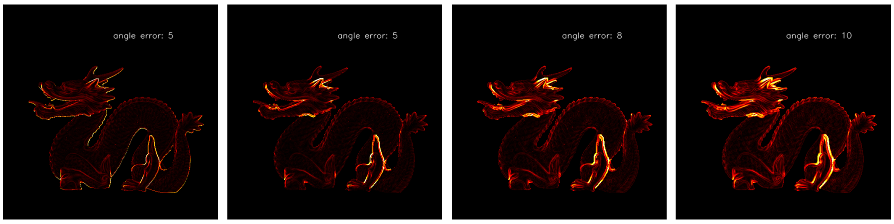


FIGURE 3.2: Error map of neighbor based method with different k values. From left to right, $k = 1, 2, 3, 4$ separately.

The performance of neighbor based method is good enough for a well chosen k. However, for the case of noised point cloud as input, this approach will broken, since the noise will failed the neighbor assumption and also reduce the number of possible neighbors of each point for a fixed k.

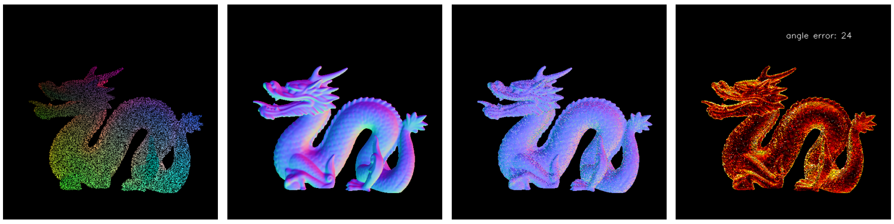


FIGURE 3.3: Evaluation of neighbor based method on a noised dragon model

3.2 Photometric Stereo

Photometric stereo was initially introduced by **photometric-stereo**, which estimates the surface normal of the object by observing the object in the same position but under different illuminated scenes. It based on the fact that the light reflected by a surface is dependent on the surface normal and the light direction.

3.2.1 Approaches

Given an image I, it can be decomposed in to two parts, the reflectance R and the shading S,

$$I = R \oplus S$$

where $\oplus$ denotes the element-wise product. This decomposition of the image is based on the intrinsic image model, which proposed by Barrow and Tenenbaum, 1978. It interprets the observed image into reflectance image and the shading image. As shown in Figure 3.4

The equation can be further decomposed based on different surface models. If assume the object surfaces are Lambertian surfaces, i.e. the surface which reflect light in all directions, the shading image can be decomposed as the product of the radiance of incoming light L_0, the cosine of the angle of incidence, which is the dot product of the surface normal N and the light source direction L.

$$I = \rho \odot (L_0 \mathbf{L} \cdot \mathbf{N})$$

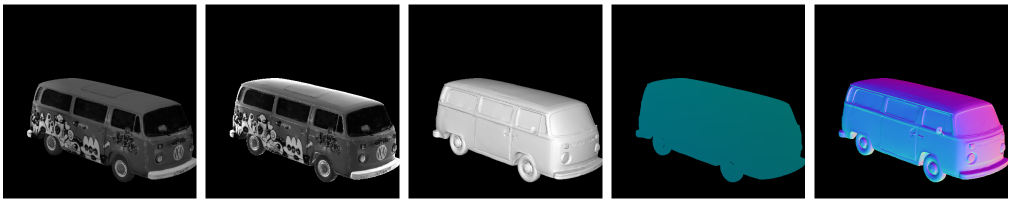


FIGURE 3.4: Intrinsic image analysis of the bus object. From left to right, original image, reflectance image, shading image, light image, normal image

note that the surface normal N and light direction L are unit vectors thus they have only two degrees of freedom.

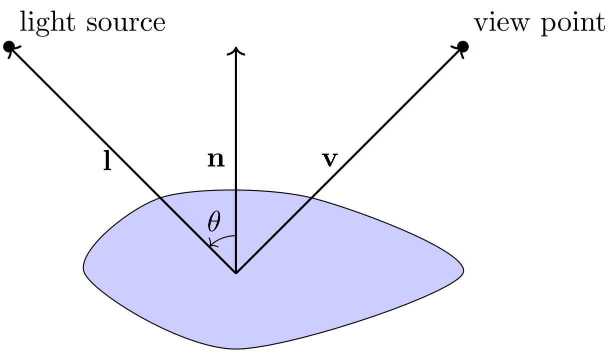


FIGURE 3.5: The surface normal, source light direction and the view point direction, where θ denotes the angle between light direction and the normal.

The equation can be further rearranged as follows

$$I = \mathbf{g} \cdot \mathbf{L} = (L_0 \rho \odot \mathbf{N}) \cdot (\mathbf{L})$$

The shape from shading method employed the equation mentioned above to predict the both surface albedo ρ and the normal $\mathbf{N}$ with knowning light source direction $\mathbf{L}$. More specifically, a set of k image for the same scene have been captured based on different light projections. Then, for each pixel (x, y) in the image, an equation system can be set up

$$\begin{pmatrix} L_1^T \\ L_2^T \\ \cdots \\ L_k^T \end{pmatrix} g(x, y) = \begin{pmatrix} I_1(x, y) \\ I_2(x, y) \\ \cdots \\ I_3(x, y) \end{pmatrix}$$

for the simplicity, L_i^T for $1 \leq i \leq k$ denotes the light direction at position (x, y) in the image k The equation can be solved based on least square methods. Since normal $N(x, y)$ is unit vector, thus we have

$$\| g(x, y) \|_2 = \| L_0 \rho(x, y) N(x, y) \|_2 = L_0 \rho(x, y)$$

Then the normal can be obtained as follow

$$N(x, y) = \frac{g(x, y)}{L_0 \rho(x, y)}$$

In another word, the surface normal including the albedo can be obtained directly based on a set of images and light directions.

3.3 Gated Convolution neural network for surface normal estimation

Recently, deep learning based method achieved a great succeed for image processing. (Redmon and Farhadi, 2018, Tan, Pang, and Le, 2019) These network architectures use a batch of RGB / Grayscale images as input and are employed for classification problems. Usually, the images are convoluted with a convolution 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.

It is worth to noticed that, the output of normal inference CNN model is not one or several labels but an entire image or normal map with same size. Recently,Ronneberger, Fischer, and Brox, 2015 proposed an architecture called UNet 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 4 times and upsampled 4 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.

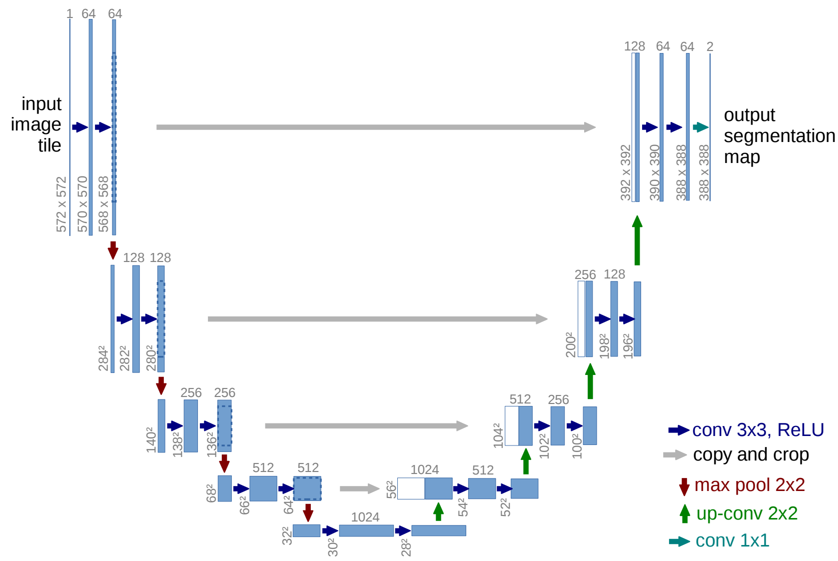


FIGURE 3.6: The structure of UNet. Ronneberger, Fischer, and Brox, 2015

The UNet is based on standard convolution layers to construct the network. This is reasonable for image processing task with full-dense input, since no missing pixels exist. However, for the input of noised point cloud, 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

$$O(x,y) = \begin{cases} \frac{\Sigma_i^k \Sigma_j^k W(i,j) \cdot I(x-i,y-j) \cdot M(x-i,y-j)}{\Sigma_i^k \Sigma_j^k W(i,j) \cdot M(x-i,y-j)}, & \text{if } \sum_i^k \sum_j^k M(i,j) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

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.

3.3.1 Gated Convolution

Oord et al., 2016 proposed a gated activation unit to model more complex interactions comparing to standard CNN layers, which mainly inspired by the multiplicative units exist in Long Short-Term Memory proposed by Hochreiter and Schmidhuber, 1997 and Rated Recurrent Unit (GRU) proposed by Cho et al., 2014. Yu et al., 2018 employed the same gated unit solving for the free-form image inpainting task. The proposed network use 3 channel RGB images as input and estimate the missing pixels.

The structure is shown in Figure 3.7. 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 $(N, C_{out}, H_{out}, W_{out})$:

$$o(N_i, C_{o_j}) = \sigma\Big(\sum_{k=0}^{C_{in}-1} w_g(C_{o_j}, k) \star i(N_i, k) + b_g(C_{o_j})\Big) * \phi\Big(\sum_{k=0}^{C_{in}-1} w_f(C_{o_j}, k) \star i(N_i, k) + b_f(C_{o_j})\Big) \tag{3.6}$$

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.

3.3.2 Architecture

Based on the implementation mentioned above, the architecture roughly follows on UNet proposed by Ronneberger, Fischer, and Brox, 2015, as shown in Figure 3.8.

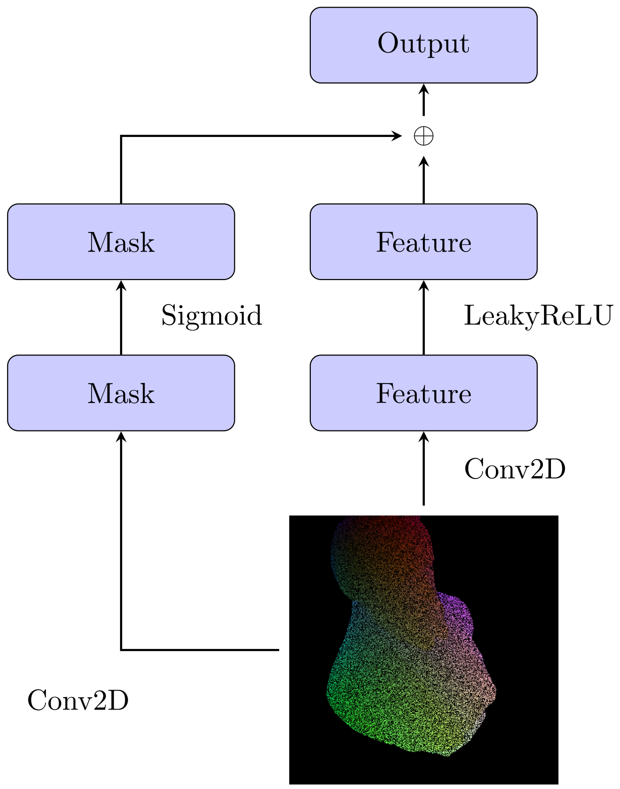


FIGURE 3.7: Gated Convolution Layer, where $\oplus$ denotes element-wise multiplication.

In order to describe the network in a common way, the parameters of the network are represented by letters. The network is constructed by a downsampling part and a upsampling part. In the downsampling part, the input is the $X \in \mathbb{R}^{w \times w \times ch}$. The input matrix goes through 3 downsampling layer blocks, each block has two gated convolution layers with stride (1,1) and an extra gated convolution layers with stride (2,2) as a downsampling operation. The total three times downsamplings extract the geometry features X_v from input matrix X (represented as a regression function f)

$$f : X \to X_f$$

After the feature extraction, the network upsampling the feature map 3 times to get the output matrix Y. Each upsampling consists of an interpolation operation uses nearest neighboring interpolations for resolution upsamling, then a concatenate layer that concatenate the interpolated result and the corresponding high resolution feature map $X_{df_1,df_2,...}$ from the downsampling part. This is also called skip connection. In the last, a gated convolution layer is utilized to reduce the channel size to fit the next upsampling block. After the three times upsampling, two standard convolution layer is added in the last without activation function to predict the surface normal. The whole upsampling branch can be represented as a regression function n,

$$n : X_v, X_{df_1,df_2,...} \to N$$

All the convolution layers in the network use same kernel size 3×3.

One of the key feature of the network is the output has the same size as the input. This is achieved by the (1,1) padding and the same channel number in the convolution layers. Thus the surface normal can be achieved 1-1 estimation. Another key point of the network is the robustness of the noise. The network takes semi-dense matrix as input then predicts the fully-dense matrix as output. The last feature is the multi-purpose using scenarios. In the description, no specific input type has been indicated. In this thesis, two application scenarios have been verified. The first is the missing-pixel estimation but with out the transformation of the style of the input matrix, which takes the semi-dense matrix as input and simply fill the missing pixels in the output. The second is the missing pixel estimation with style-transferred of the matrix. In this case, the network takes the semi-dense matrix as input, the output matrix is fully-dense but each pixel has the different meaning as the input. Actually, the first scenario can be consider as the specially case of the second scenario which the style of output and the input remain the same. The network is test in Chapter 5, which is shown the good performance on noise filtering task and also the normal surface estimation task.

3.3.3 Loss Function

L1 Loss L1 loss, also known as absolute error loss, which calculates the absolute difference between the prediction and the ground truth. It leads to the median of the observations.

$$L_1(\tilde{y} - y) = |\tilde{y} - y|$$

L2 Loss The standard loss function for optimization in regression problems is the L2 loss, also known as squared error loss, which minimize the squared difference between a prediction and the actual value. It leads to the mean of the observations.

$$L_2(\tilde{y} - y) = \|\tilde{y} - y\|_2^2$$

Masked L2 Loss with penalty for outliers(mask-l2) The background pixels of the input data are not considered in the normal inference task, they are saved as black pixels in the input data. These pixels should not considered in the loss function, i.e. invalid pixels. Therefore, a valid mask is required to distinguish the background and the main object. Specifically, using a matrix with the same width and height as the output, for each pixel, 0 is invalid, 1 is valid. Furthermore, depends on the specific task, the output should be constraint in a range. For normal output, the range is $[-1, 1]$. Thus for the outliers out of this range, a outlier mask can be applied to give them a penalty.

$$\begin{aligned} l(x, y) &= L = \{l_1, ..., l_N\}^T \\ l_{n \in N} &= \|mask_{obj} \odot mask_{ol} \odot (\tilde{y}_n - y_n)\|_2^2 + \|mask_{obj} \odot mask_{nol} \odot (\tilde{y}_n - y_n)\|_2^2 \end{aligned} \tag{3.7}$$

where x is input, y is target, N is the batch size.$mask_{obj}$ is the mask of the object, i.e. 1 means it is an pixel on the object, 0 is an pixel on the background. $mask_{ol}$ is the mask for the outliers, i.e. 1 means outliers, 0 means non outlier, $mask_{nol}$ is exactly the inverse of $mask_{ol}$. p is the penalty of the outliers, it is set as 1.4.

Reversed Huber Loss Owen, 2007 proposed Reversed Huber loss to combine both L1 and L2 loss. L1 loss is for small values whereas L2 for large values

$$\mathcal{B}(y) = \begin{cases} |y| & |y| \leq c \\ \dfrac{y^2 + c^2}{2d} & |x| > c \end{cases} \tag{3.8}$$

where $c = 0.2 \max(|\tilde{y} - y|)$.

3.4 Guided normal inference using GCNN

The guided normal inference takes the light direction and the image into consideration. It is under the assumption that the scene image I is captured by a calibrated camera, i.e. knowing the intrinsic K and extrinsic camera matrix $[R|t]$, and a the light position (s_x, s_y, s_z) of the single light source. The geometry based approach inference the surface normal from the point cloud, whereas the photometric stereo inference the surface normal from a set of calibrated illuminated images. The idea behind this chapter is that improve the geometry based approach with the help of one calibrated illuminated image. Since the calibrated illuminated image contains the information about the surface feature, it is supposed to help the geometry approach in a proper way. Based on this idea, two networks are proposed in this section.

3.4.1 Light Map

The light map L can be derived from vertex map V and the light source position s. As shown in Figure 3.5, the incoming light direction is a vector point from light source to the surface point, therefor it can be calculated as follows

$$L(x, y) \quad = \frac{V(x, y) - (s_x, s_y)}{\|V(x, y) - (s_x, s_y)\|_2} \tag{3.9}$$

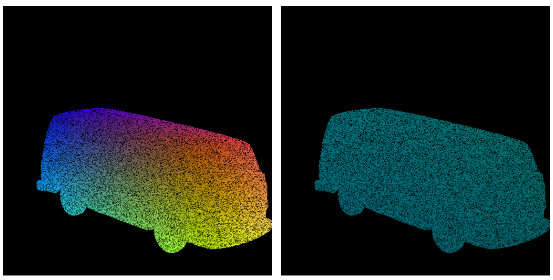


FIGURE 3.9: The light map calculated from vertex map and the light source

where (s_x, s_y) is the light source position and V is the vertices, both (s_x, s_y) and V are with respect to the camera space. The light direction map L is normalized since only the direction of the light is considered. Using the equation above for all the pixels in the point cloud can obtain the corresponding light map, which is a matrix with same size as point clouds. However, it is important to note that due to the exist noise in the vertex map, the getting light map is only semi-dense, as shown in Figure 3.9.

3.4.2 VIL Net

Based on above implementations, we propose a light and image guided network called Vertex-Image-Light Network (VIL-Net). The structure is basically derived from GCNN model as mentioned in 3.3, which is shown in Figure 3.10.

As mentioned in the name, the **VIL**-Net utilizes **V**ertex map, **L**ight map and **I**mage map to accomplish the normal inference task.

The network can be consider in two parts. The first part extracts the feature maps from the input data. It deals with two kinds of input, the vertex map $X_1 \in \mathbb{R}^{w \times h \times 3}$, and the concatenation of light and image map $X_2 \in \mathbb{R}^{w \times h \times 4}$. The network extracts the geometry features X_v from vertex map X_1 (represented as a regression function v)

$$v : X_1 \to X_v$$

and the photometric features X_l from image and the light map X_2 (represented as a regression function l)

$$l : X_2 \to X_l$$

where the two encoders have the same network architecture based on the downsampling part of GCNN model. After the feature extraction, 2 extra layers are added: 1, a concatenate layer is added to fuse the vertex feature, and the image and light map feature getting from the encoder. 2, a fused feature map is predicted from all the feature maps base on a single gated convolution layer. (represented as a regression function m)

$$m : [X_v X_l] \to X_f$$

Then the network interpolates the feature maps X_f 3 times using interpolation and gated convolution layers to inference the normal map N. Meanwhile, the skip connections fuse the high resolution features $X_{df_1, df_2, \ldots}$ from the downsampling part during the upsamplings. The upsampling is represented as a regression function n:

$$n : X_f, X_{df_1, df_2, \ldots} \to N$$

With the help of an extra image-light encoder, the network gained more information of the object surface, which is supposed to predict the surface normal more accurate. In this scenario, the output is still the surface normal, thus the training loss can be the same as GCNN model.

3.4.3 TriG-Net

We propose a light and image guided network called Triple-pipe-gated-Network (Trip-Net), which employs the GCNN architecture three times to accomplish the normal inference task. The architecture is shown in Figure 3.11.

The network has three pipes combined with one main pipe and two side pipes. Each pipe deals with different task. The main pipe deals with the geometry information, which takes the vertex map as the input and used to predict the surface normal. The light map is fed into a side pipe and used to extract the light feature. The network forwards the features to the main pipe as a supplementary information for the normal estimation. The image pipe takes the image as the input and extracts the image features then forward the features to the main pipe as well. The supplementary pipes provide the illumination information which helps the main pipe to refine the inferred normals.

Side-Pipe(Light)

The task of light pipe in the network is to predict the light feature from the photometric information using the light direction in the scene, which is calculated from the point cloud and the light position. The light map also inherits the noise from the point cloud, which lead to a semi-dense input map. Therefore this side pipe utilizes gated convolution layers for feature map extraction to handle the missing values. The input vertex map is downsampled three times in this part. Each downsampling utilizes three gated layers, the first two layers have stride (1,1) and the third one has stride (2,2) to reduce the feature scale. In the upsampling part, the network also has three times upsampling as a symmetric design. In each upsampling, it first uses nearest neighbor interpolation algorithms to expend the feature map size. Then it concatenates the corresponding feature map from the downsampling part. In the last, the feature map goes through a gated convolution layer to reduce the channel size.

Side-Pipe(Image)

The task of image pipe in the network is to predict the image feature. This pipe is a collaborate pipe with the light pipe. The architect is the same as light map pipe but only the input is the image matrix.

Main-Pipe(Vertex)

The task of vertex pipe in the network is to predict the normal map directly. The input is vertex map converted from point cloud. The downsampling part is still the same as the other two pipes. The different part is the upsampling, which has to consider the information coming from the other two pipes. After the downsampling, a concatenation layer fused the corresponding feature map from the other two pipes, then it takes 3 times upsampling. Each upsampling consists of 5 layers: 1, a interpolation layer to double size the resolution using nearest neighboring interpolation algorithm, 2, a gated layer to reduce the channel size to 1/3 of itself in order to fit the corresponding feature map in the downsampling part, 3, a concatenation layer to fuse the output with the corresponding feature map in the downsampling part, 4, a gated layer to reduce the channel size to 1/2, 5, a concatenation layer to fuse the corresponding upsamling feature map from the other two pipe altogether with the feature map in current pipe. These 5 layers consider both the information from other pipes and also keeps the high resolution from the downsampling part. A gated convolution layer is added afterwards to reduce the channel size and extra two standard convolution layers are used to predict the final surface normal map.

3.4.4 Light and image guided normal inference using GCNN

Based on above implementations, we propose a light and image guided network called Vertex-Image-Light Network (VIL-Net). The structure is shown in Figure 3.12. As mentioned in the name, the network utilizes the GCNN architecture three times to accomplish the task. The first branch (shown above) takes a light map introduced in 3.4.1 as the input, the structure is the same as GCNN architecture except that the last two standard convolution layers, the skip connections are kept to connect the 3 down/up samplings. The second branch (shown below) takes image as the input, the structure is the same as the first branch other than the input image is 1 channel but not 3 channels. The third branch takes the 3D vertex map as the input. The structure is based on GCNN architecture. However, in order to merge the other two branches, the vertex branch equips 4 times fusions in the up sampling part. Specifically, the first fusion locates immediately after the last gconv layer of the last down sampling, the second fusion after the second gconv layer of first up sampling, the third fusion after the second gconv layer of second up sampling, the fourth fusion after the second gconv layer of the third up sampling. Each fusion follows by an interpolation layer, a gconv layer to reduce the channel back to 32, a skip connection concatenate layer and another gconv layer to reduce the channel back to 32. After the fourth fusion, a gconv layer used for channel reduction, 2 standard conv layer for output prediction.

3.4.5 Loss Function

For the case of normal output, the loss function is the same as Mask-L2 loss as introduced in 3.3.3. For the case of the product of albedo and normal, the loss function utilized a scaled Mask-L2 loss, which gives the range of inliers between $[0, 255]$.

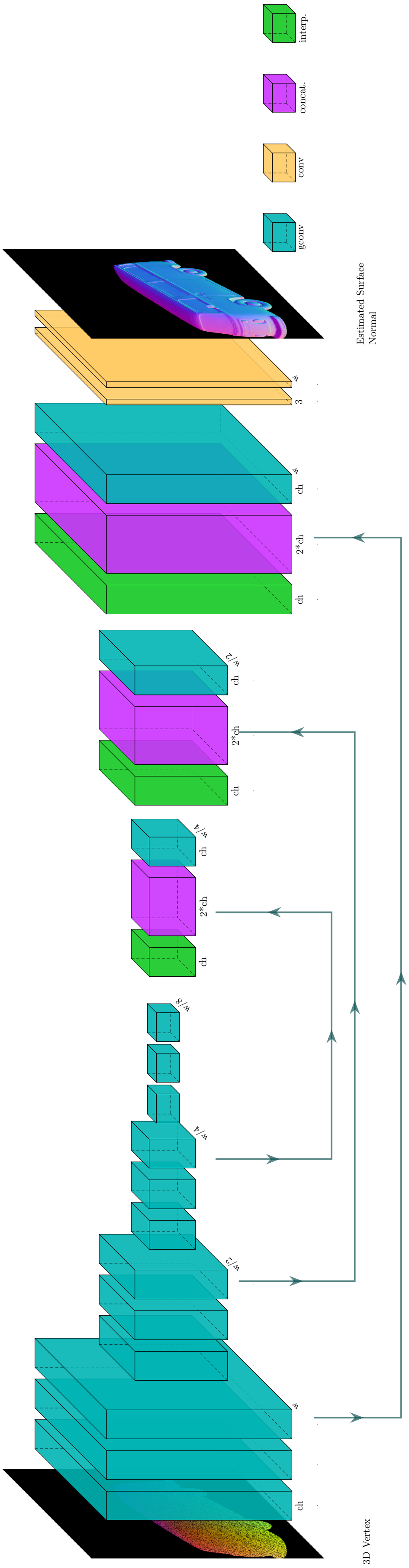


FIGURE 3.8: The architecture of Gated convolution neural network (GCNN) based on Gated convolution and UNet Architecture.

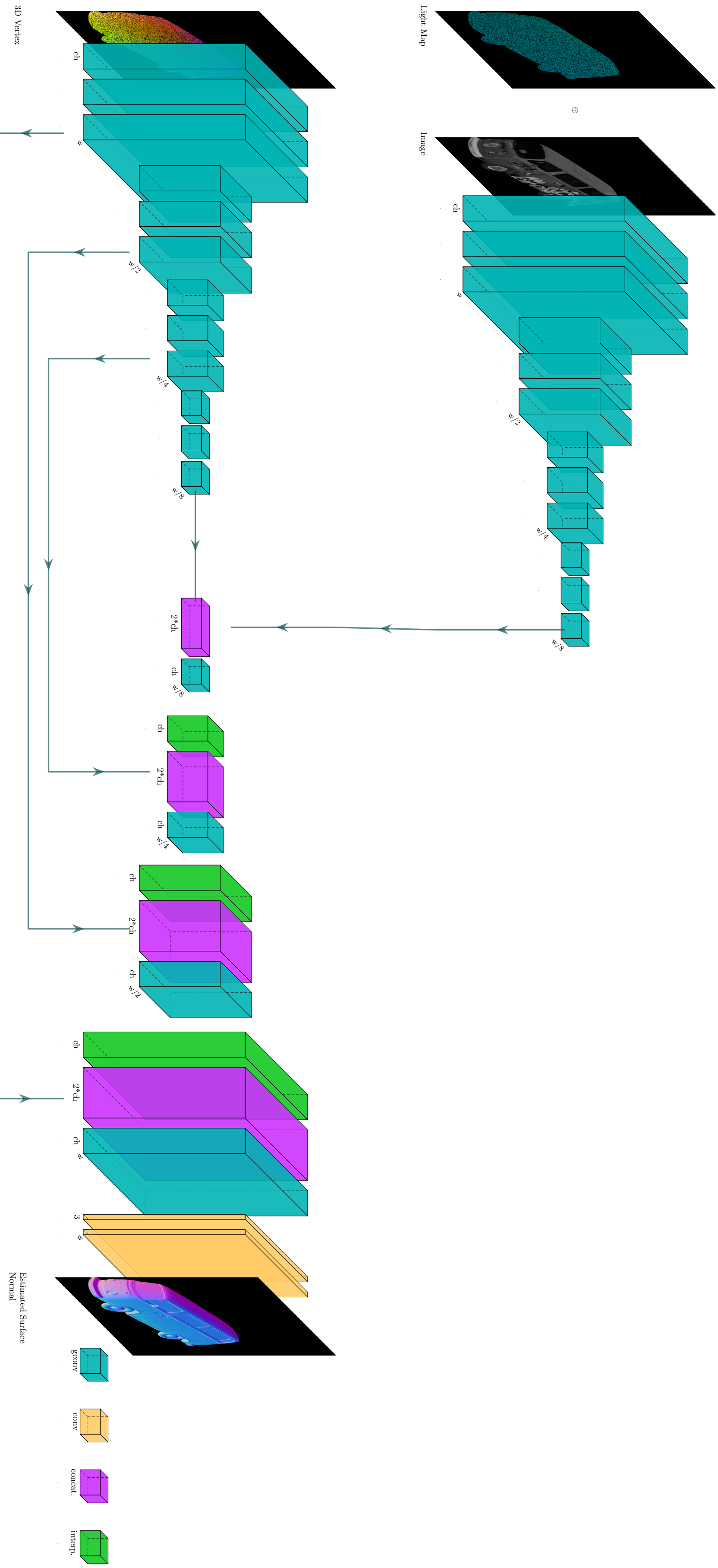


FIGURE 3.10: The architecture of VIL-Net

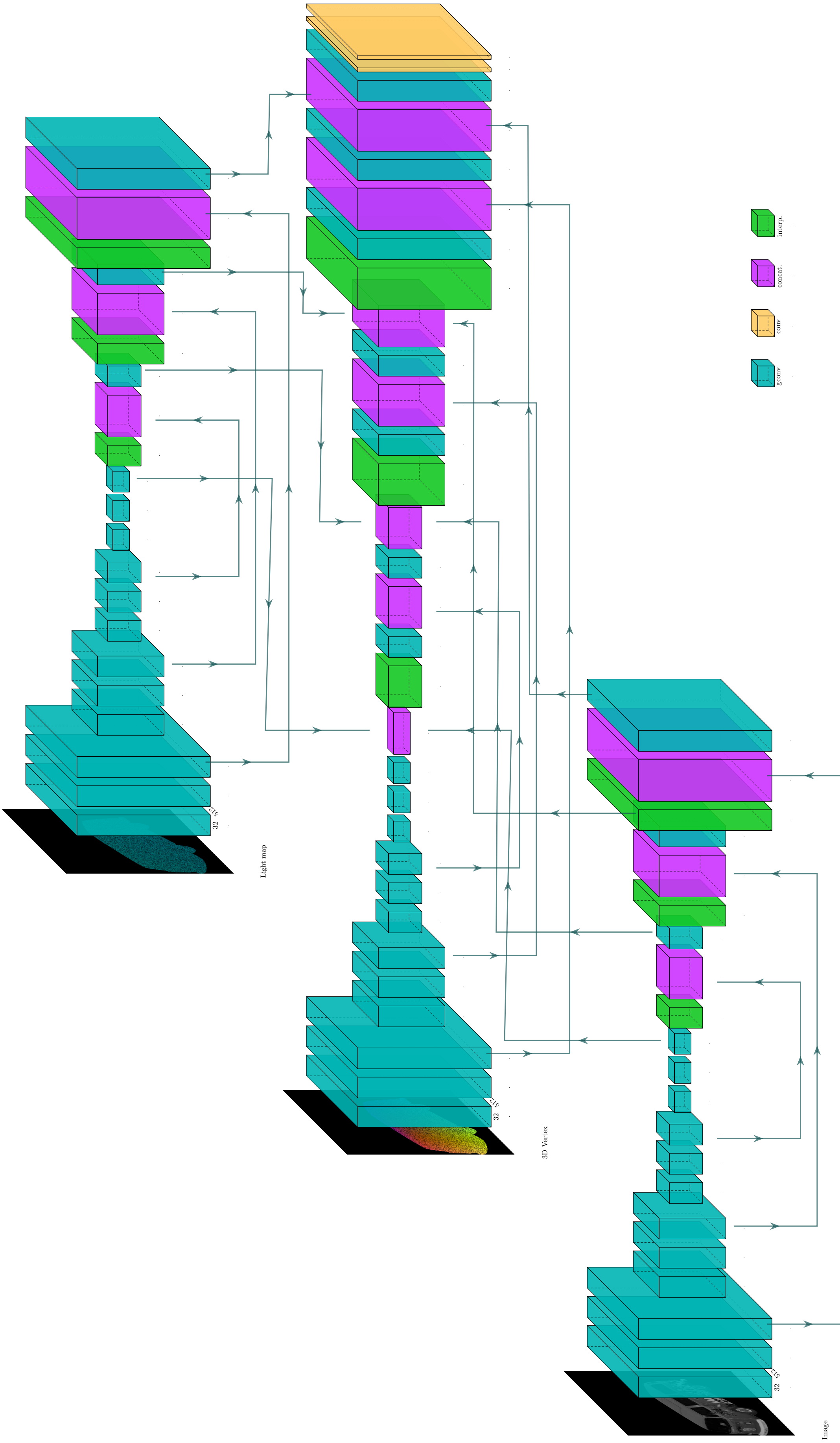


FIGURE 3.11: The architecture of Trig-Net

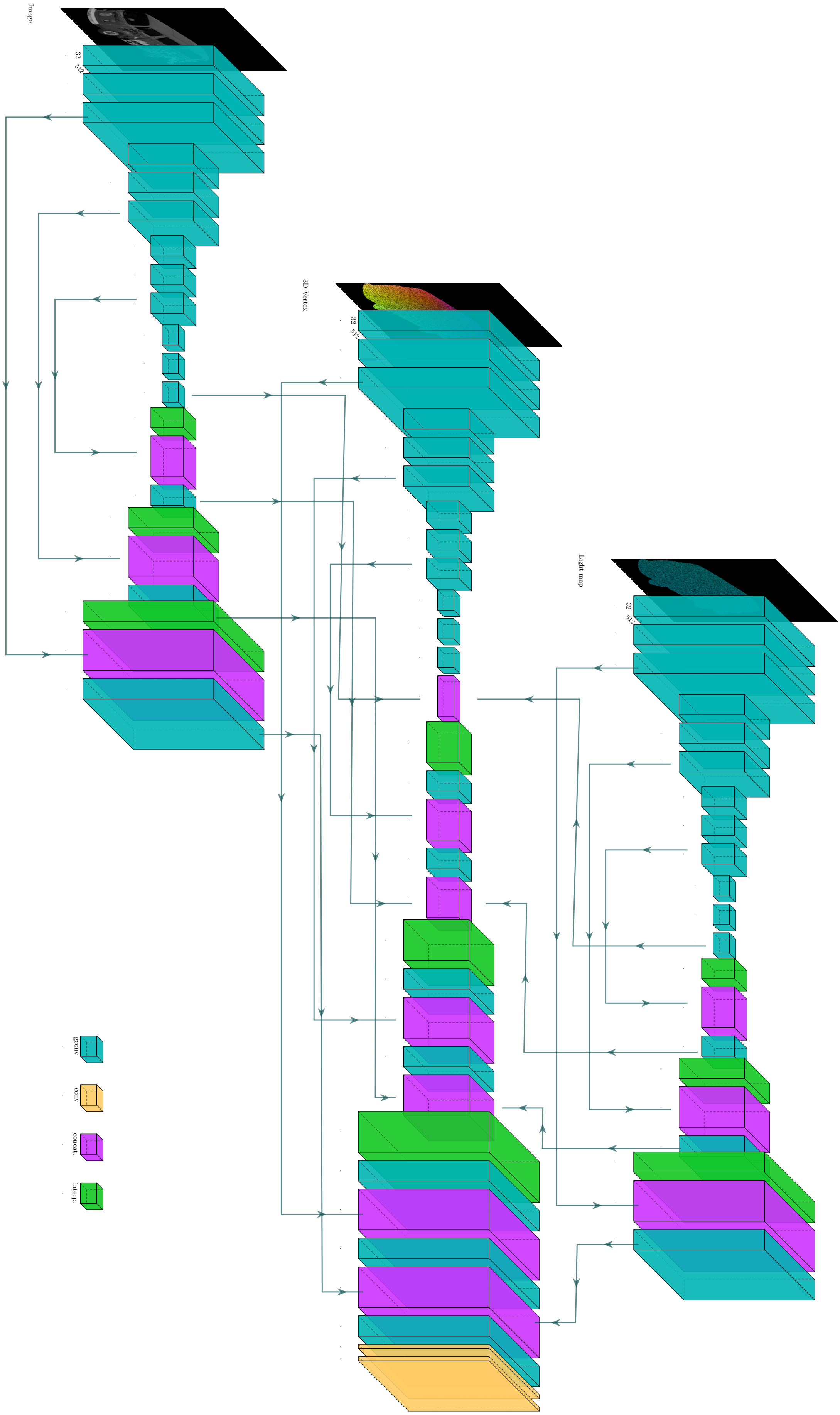


Figure 3.12: The architecture of TriGNet

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 **??**. 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.

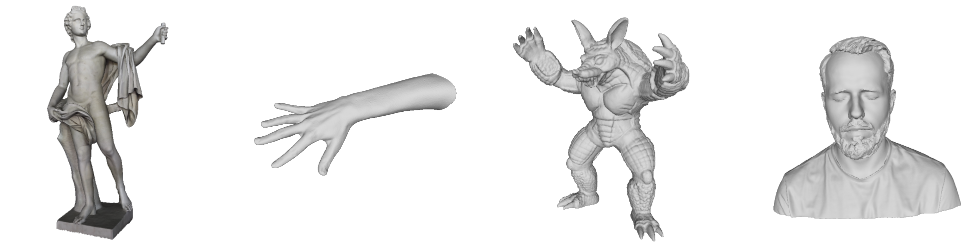
The "synthetic50-5", is a datset with 50 point clouds as training set and 5 point clouds as test set. The dataset is created base on the work of this thesis and using for normal inference task. Figure **??** gives the illustrations of some objects. Appendix A gives a full version of dataset models.

4.1.2 Synthesize Scenes using Unity

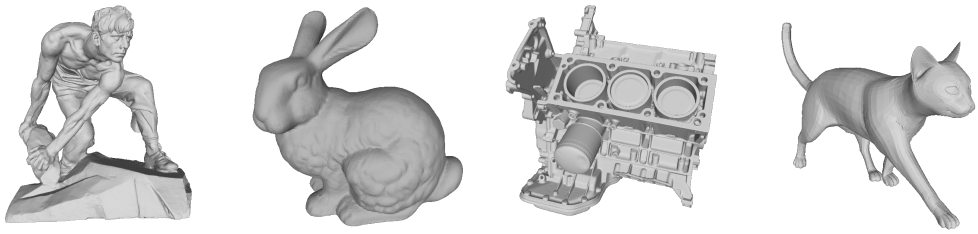
In order to fit the using scenario of Kinect as much as possible, the dataset consists of the generated synthetic 3D scenes via Unity, which is a game engine using for 3D games creation.

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

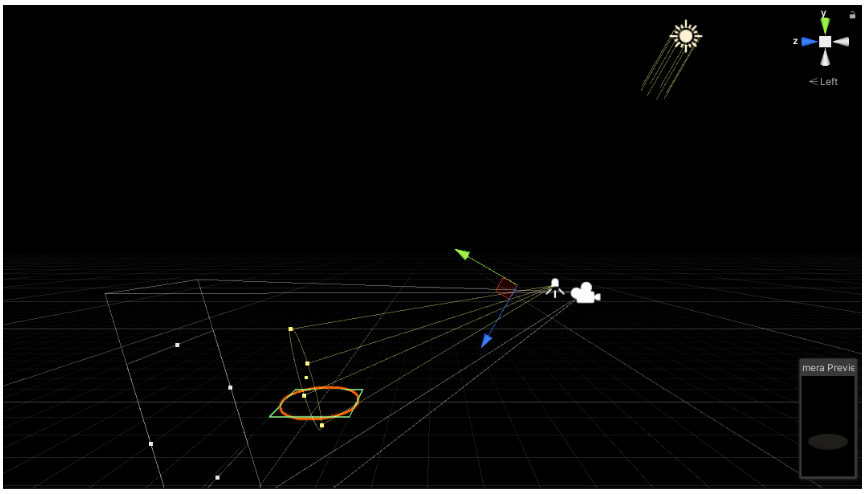


FIGURE 4.2: The layout of synthetic scenes generation in Unity.

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:

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

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.

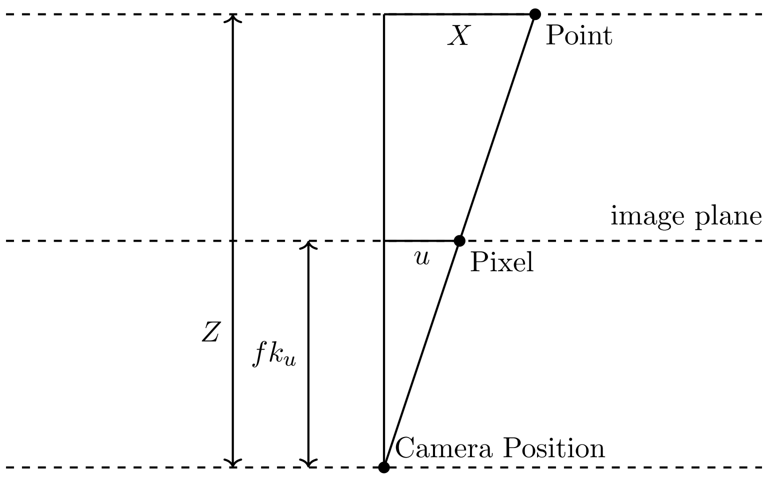


FIGURE 4.3: Convert depth to point in camera coordinate system

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.

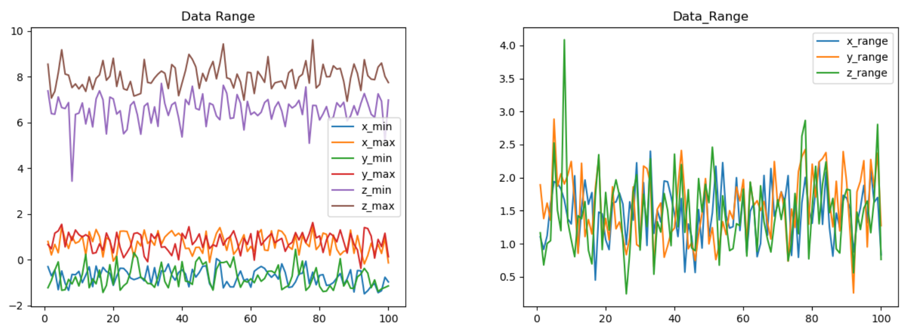


FIGURE 4.4: Left: Extreme value in 3 axis; Right: Vertex range in 3 axis

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.

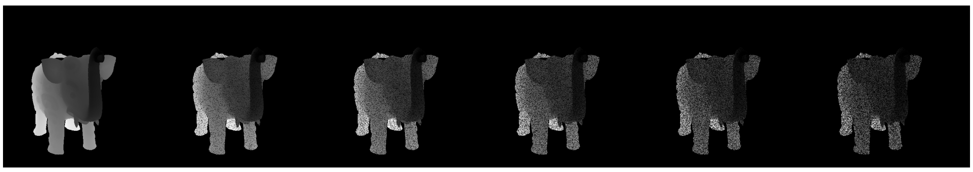


FIGURE 4.5: Noise-intensity on $\mu{-}0, \mu{-}10, \mu{-}20, \mu{-}30, \mu{-}40, \mu{-}50$,. Object Name: elephant-zun-lid.

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.

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 1.10.0a0, CUDA 11.4.1, GPU with single NVIDIA GEFORCE RTX 3090.

5.1 GCNN model evaluation

In order to ease this issue, a light inpainting model has been trained based on the GCNN network with the identity architecture but the input and output, which takes the semi-dense light map as input and predict the fully dense light map as output.

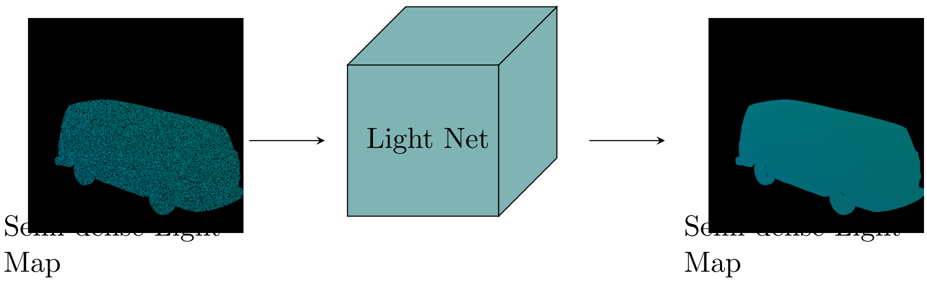


FIGURE 5.1: Light Net for light inpainting based on GCNN architecture.

5.2 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. We evaluate the performance of the model in dataset "synthetic50-5" introduced in 4 including 5K depth maps with size $512 \times 512 \times 3$ for training. All the depth maps add simulated noise as introduced in 4.1.5. The training pipeline use batch size 8, Adam optimizer (Kingma and Ba, 2014), learning rate of 1×10^{-3}, penalty-l2 loss. The output is directly the tangent normal in range $[-1, 1]$. The output and input has the same shape.

5.2.1 Qualitative Evaluation

The evaluation visualization on test dataset is shown in Figure 5.2. As shown in the picture, GCNN model Figure 5.3 zooms in the hindhead area of the dragon object which provides a closer comparison with the ground-truth.

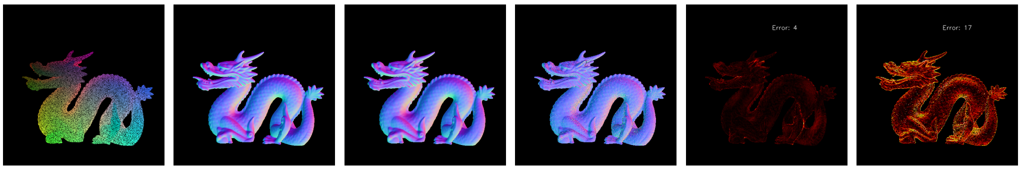
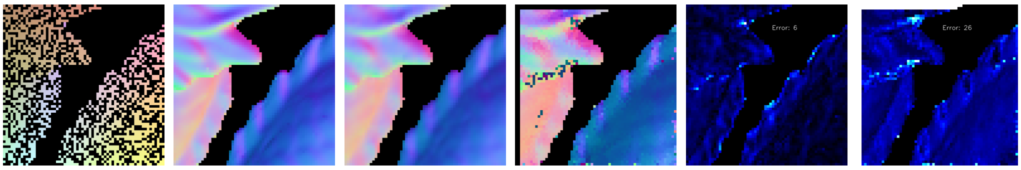


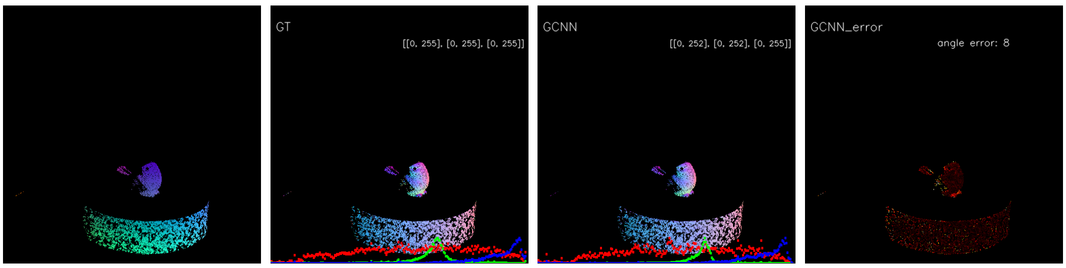
FIGURE 5.2: GCNN Normal Inference on Synthetic Dataset (object: dragon) From left to right: Input vertex map, ground-truth, GCNN normal map, SVD normal map, GCNN error map, SVD error map



(A) Input (B) GT (C) GCNN (D) SVD (E) (F)

FIGURE 5.3: Zoom in of dragon object evaluation.

The evaluation visualization on real dataset is shown in Figure 5.4



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

FIGURE 5.4: Evaluation on Real Dataset

5.2.2 Quantitative Evaluation

The evaluation result on test scenes is shown in Figure 5.5. 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, 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.

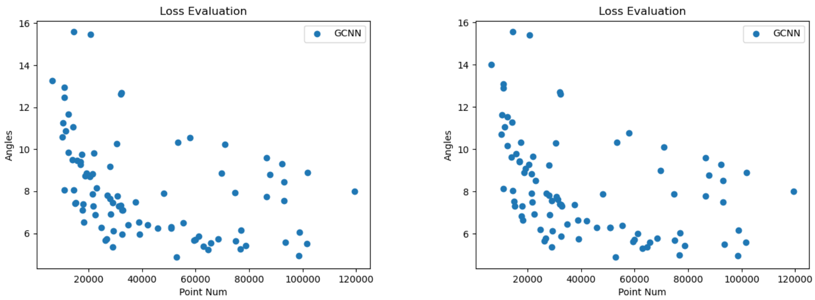


FIGURE 5.5: Evaluation of average angular loss on the whole test dataset with 90 scenes. The x-axis indicates the point number, the y-axis indicates the angles. The **Left** one using point cloud without noise, the **right** one has noise.

5.2.3 Speed

5.3 Guided Gated Convolution Neural Network for Normal Inference

5.3.1 Light Net Evaluation

We evaluate the GCNN architecture based light net on the test dataset. The light net parameters are further used as the initial parameter of light branch of the Trignet.

A qualitative evaluation of light net is shown in Figure 5.6. The average angular error are lower than 0.3 degree on all of test cases. In this case, to distinguish the output and the ground truth is already not easily only use naked eyes.

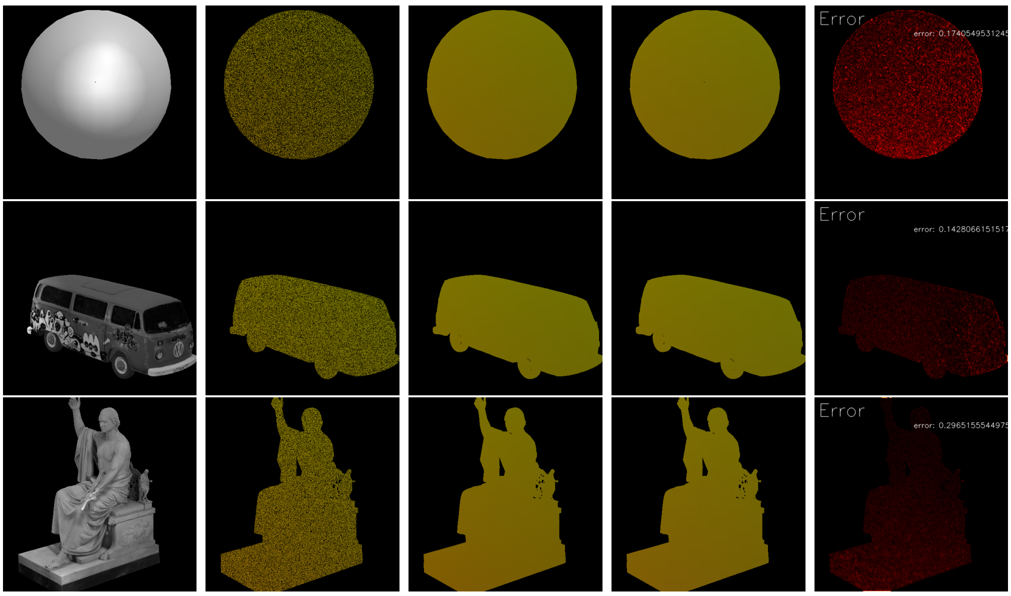


FIGURE 5.6: Qualitative evaluation of light net on Sphere, bus and Washington statue. From left to right, grayscale image, semi-dense light map(input), full-dense light map(output), full-dense light map(ground-truth), error map.

Figure 5.7 uses a scatter plot to show the average angular error on 100 different test cases. The average pixel-wise angular error is 0.17 degree as shown in Table 5.1. An regression line has been added in the plot to analysis the tendency of the errors.

The angular error slightly goes up when valid point number in the scene increase. It is make sense since the valid pixels are usually connected and concentrate in a single patch, the less of the area of the patch, which corresponding the number of the valid points, the less variation of the light direction among the pixels, thus better the evaluation.

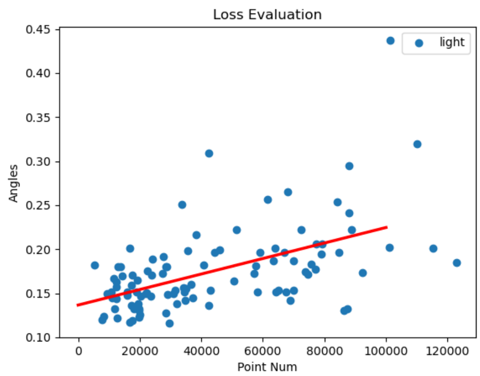


FIGURE 5.7: The evaluation on 100 test cases of Light Net.

5.3.2 Vertex-Image-Light Net (VIL Net) Evaluation

The qualitative evaluation of VIL net uses the model trained based on masked L2 loss.

title

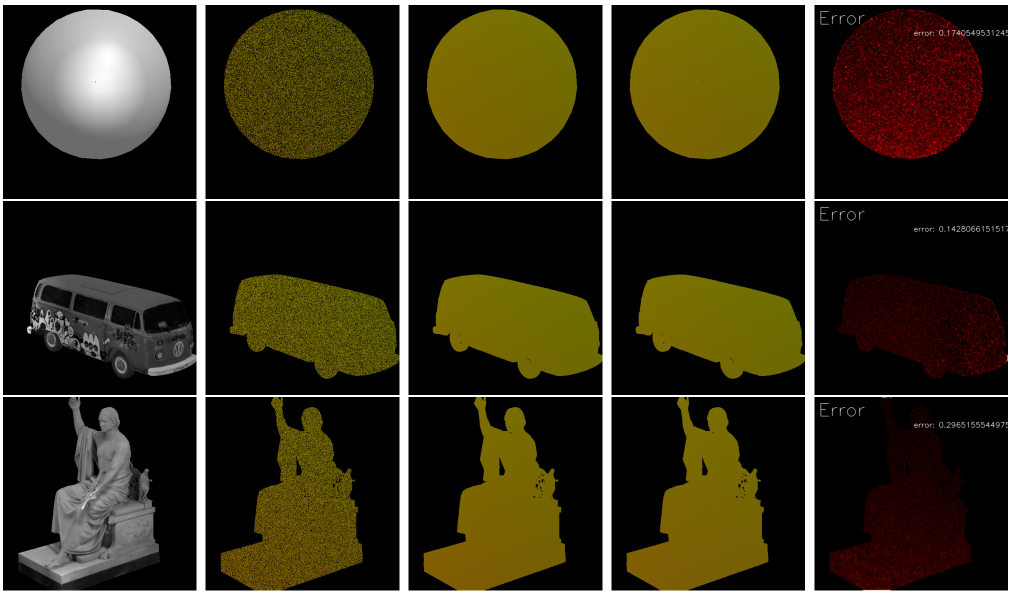


FIGURE 5.8: Qualitative evaluation of Trignet net on Sphere, bus and Washington statue. From left to right, semi 3D vertex map, full-dense normal map(output), full-dense normal map(ground-truth), error map.

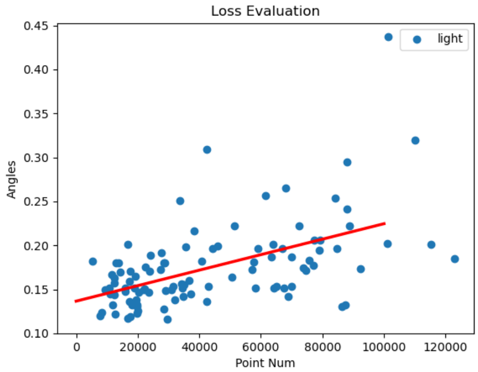


FIGURE 5.9: The evaluation on 100 test cases of Trignet.

5.3.3 comparison

From the Figure 5.11 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 quite 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?

Model	**Angle**	**Time /ms**	**bz**	**lr-schedule**	**lr-df**	**l/i. Nr.**	over-perform
LightNet	0.17	4.72					
GCNN	10.57	5.25	8	200,1600	0.5	0	-
an3	10.46	64.86	8	3,12,1000	0.5	1	yes
an3	10.81	65.34	8	10,1000	0.5	1	no(yes)
VIL-1	10.50	31.57	8	10,1000	0.5	1	yes
VIL-1	10.82	32.61	8	3,12,1000	0.5	1	no(yes)
VIL-3	10.79	54.76	8	100,1000	0.1	3	no(yes)
VIL-10	11.10	132.32	8	100,1000	0.1	10	no(yes)

TABLE 5.1: The error of models. The angle error is the average angle error of all valid pixels in the test case. The time unit is millisecond. bz is the batch size, lr-schedule is learning rate schedule. lr-df is learning rate decay factor, l/i. Nr is the number of light-image maps used for each scene

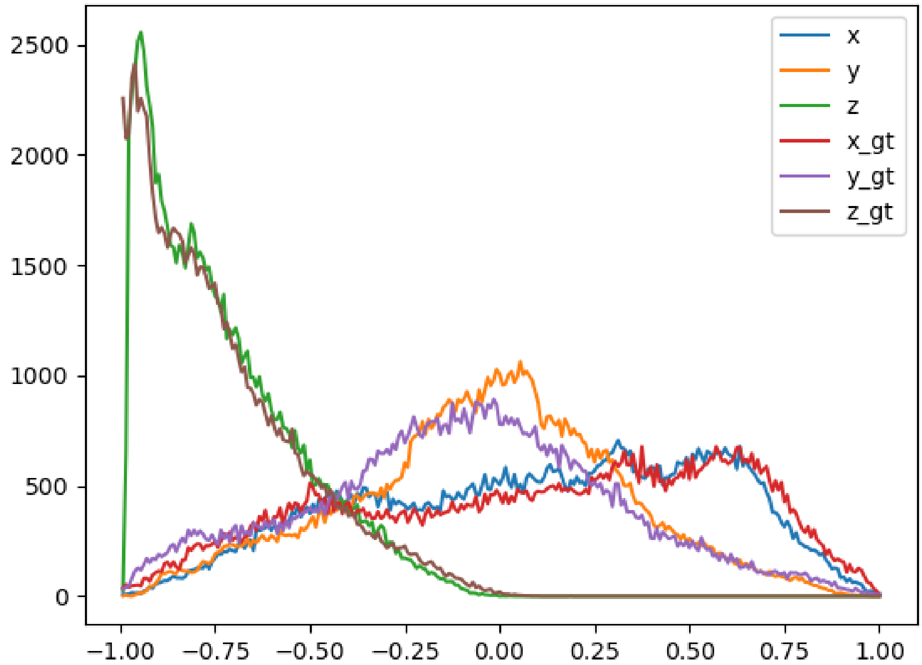


FIGURE 5.10: The normal difference of between GCNN and ground-truth in x, y, z-axis respectively. The y axis indicates the number of points, x axis indicates the value of normal in x/y/z axis. (The chart is based on the ”dragon” scene showing above)

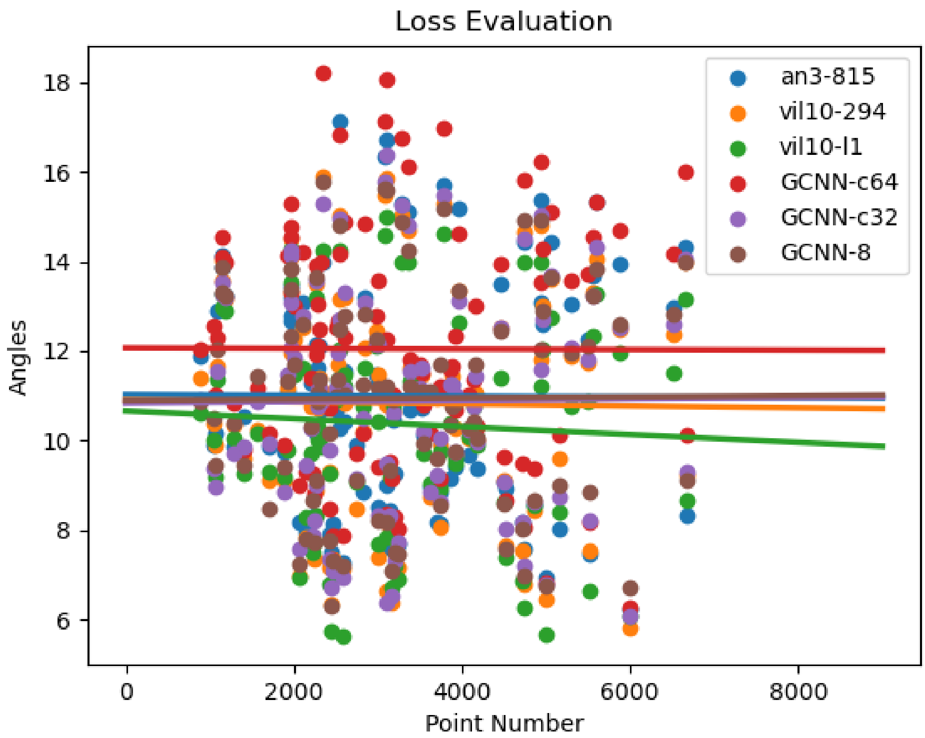


FIGURE 5.11: The normal difference of between GCNN and ground-truth in x, y, z-axis respectively. The y axis indicates the number of points, x axis indicates the value of normal in x/y/z axis. (The chart is based on the "dragon" scene showing above)

Chapter 6

Conclusion

Gated convolution neural network...

Appendix A

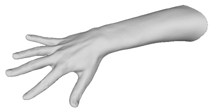
Dataset

A.1 Dataset

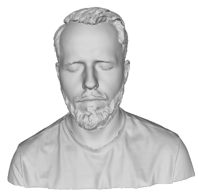
A.2 How do I change the colors of links?

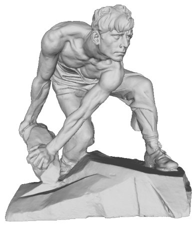
The color of links can be changed to your liking using:
`\hypersetup{urlcolor=red}`, or
`\hypersetup{citecolor=green}`, or
`\hypersetup{allcolor=blue}`.
If you want to completely hide the links, you can use:
`\hypersetup{allcolors=.}`, or even better:
`\hypersetup{hidelinks}`.
If you want to have obvious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

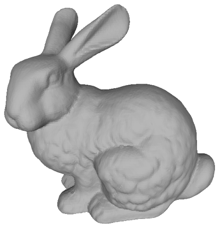

(A) apoll

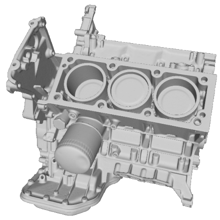

(B) arm

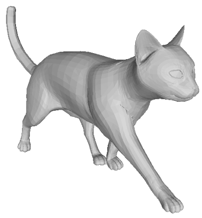

(C) armadillo


(D) bearded-guy


(E) bronze-sculpture


(F) bunny


(G) car-engine


(H) cat

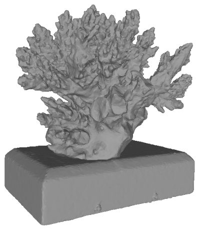

(I) side-chair

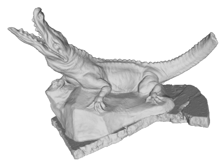

(J) chair2


(K) chandelier


(L) Christmas-bear


(M) classic-side-table


(N) coral

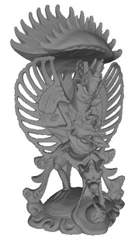

(O) crocodile-statue

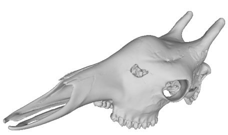

(P) Diane


(Q) Dresdner-knabe


(R) fantasy-dragon


(S) fox-skull


(T) Garuda and Vishnu

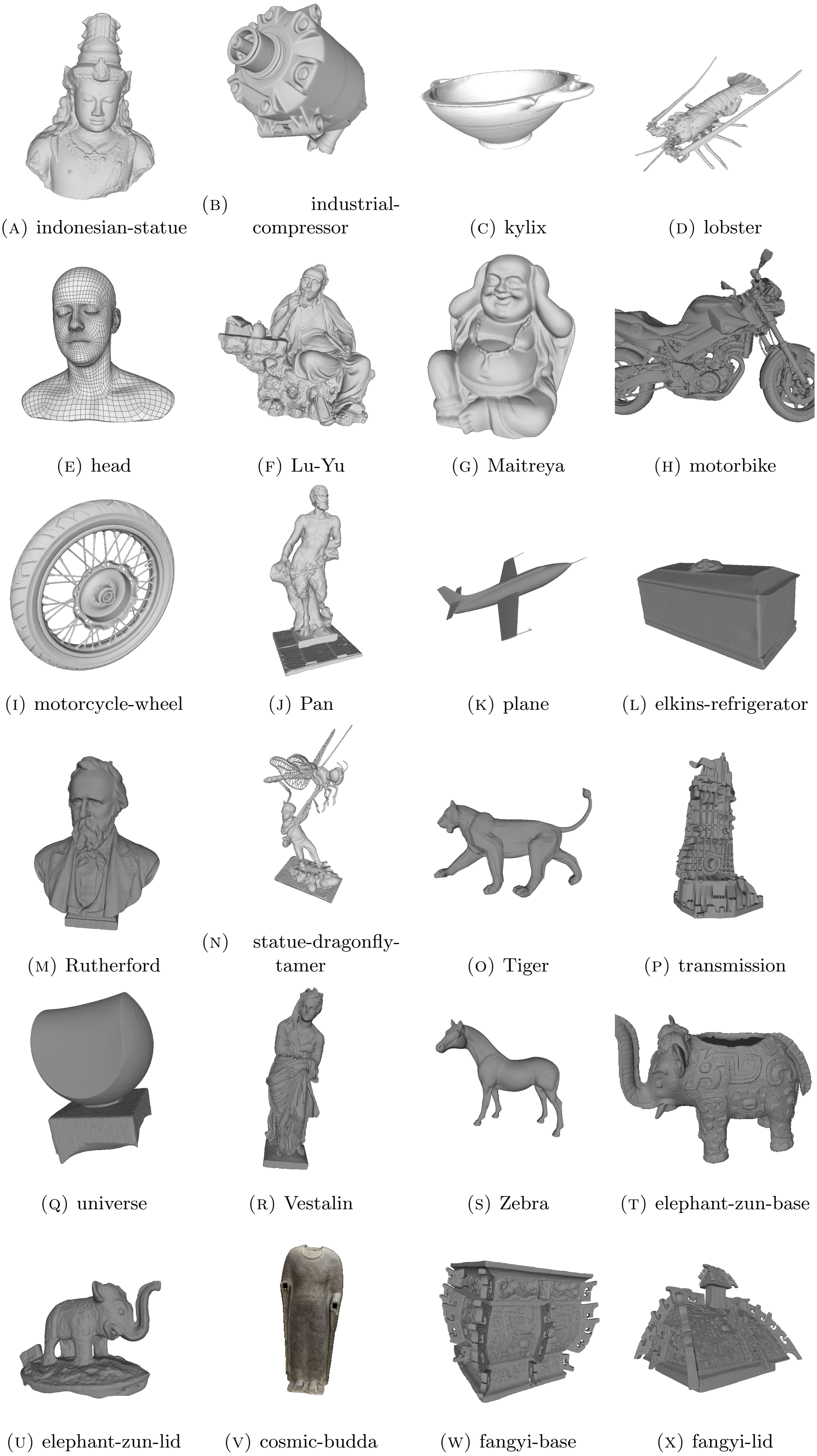

(U) giraffe-skull


(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

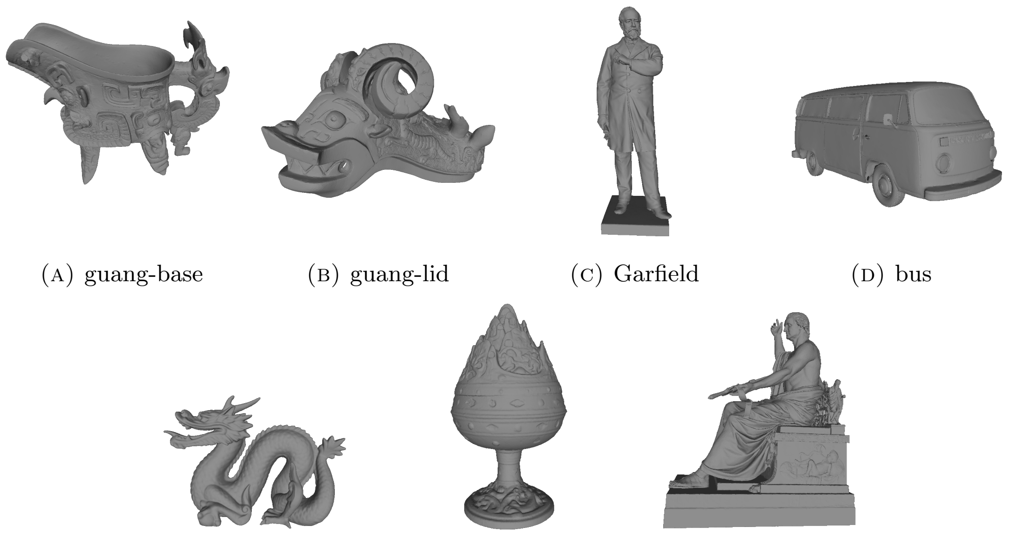
(I) motorcycle-wheel (J) Pan (K) plane (L) elkins-refrigerator

(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

Barrow, Harry and J. Tenenbaum (Jan. 1978). "Recovering Intrinsic Scene Characteristics from Images". In: *Recovering Intrinsic Scene Characteristics from Images.*

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.

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.

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.

Klasing, Klaas et al. (2009). "Comparison of surface normal estimation methods for range sensing applications". In: *2009 IEEE International Conference on Robotics and Automation*, pp. 3206–3211. DOI: 10.1109/ROBOT.2009.5152493.

Knutsson, H. and C.-F. Westin (1993). "Normalized and differential convolution". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 515–523. DOI: 10.1109/CVPR.1993.341081.

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.

Li, Zhenyu et al. (2022). *BinsFormer: Revisiting Adaptive Bins for Monocular Depth Estimation*. DOI: 10.48550/ARXIV.2204.00987. URL: https://arxiv.org/abs/2204.00987.

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.

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.

Owen, Art (Jan. 2007). "A robust hybrid of lasso and ridge regression". In: *Contemp. Math.* 443. DOI: 10.1090/conm/443/08555.

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].

Silberman, Nathan et al. (2012). "Indoor Segmentation and Support Inference from RGBD Images". In: *ECCV'12 Proceedings of the 12th European conference on Computer Vision - Volume Part V*. Springer-Verlag Berlin, pp. 746–760. ISBN: 978-3-642-33714-7. URL: https://www.microsoft.com/en-us/research/publication/indoor-segmentation-support-inference-rgbd-images/.

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/.

Uhrig, Jonas et al. (2017). "Sparsity Invariant CNNs". In: *International Conference on 3D Vision (3DV)*.

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.

Zeng, Jin et al. (2019). "Deep Surface Normal Estimation With Hierarchical RGB-D Fusion". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6146–6155. DOI: 10.1109/CVPR.2019.00631.

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