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

List of Symbols

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

To ...

Chapter 1

Introduction

1.1 From 3D to 2D

1.2 Kinect and Surface Normal

As a popular and affordable new type of depth sensor, Kinect, had attracted a great focus to computer vision research in the last decades. It simultaneously captured the grayscale or RGB image and depth map with a high resolution. The depth map can further convert to structured point clouds with known intrinsic calibration parameters, which can use in many applications.

Surface Normal is one of the most worthy pieces of information that can infer from point clouds, which applies in many practical applications, such as augmented reality and robotics. Some tasks like shading and surface reconstruction require normal as one of the inputs. However, due to the lack of accuracy of the sensors, the surface normal inference from depth maps/point clouds has many challenges.

1.3 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.

1.4 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 down-sampling 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.

1.5 Challenges of Normal Inference

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.

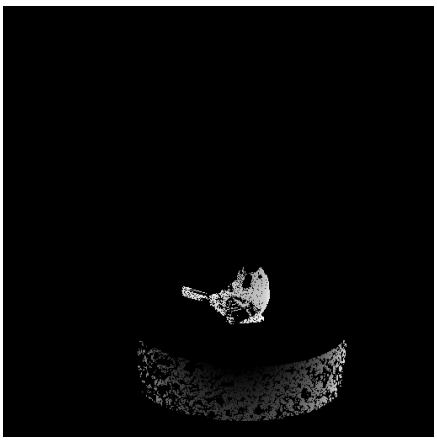


FIGURE 1.1: A captured depth map via infrared sensors. Pixels that far away represent by light colors, otherwise by dark colors. The black dots are the depths that failed to be detected.

1.6 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 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.

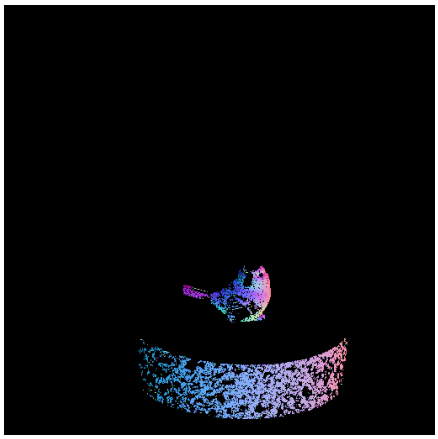


FIGURE 1.2: Semi-dense Normal Map calculated from depth map using a standard method

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.

Chapter 2

Related Work

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.

2.1 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.

2.2 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

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 $(N, C_{out}, H_{out}, W_{out})$:

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

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.

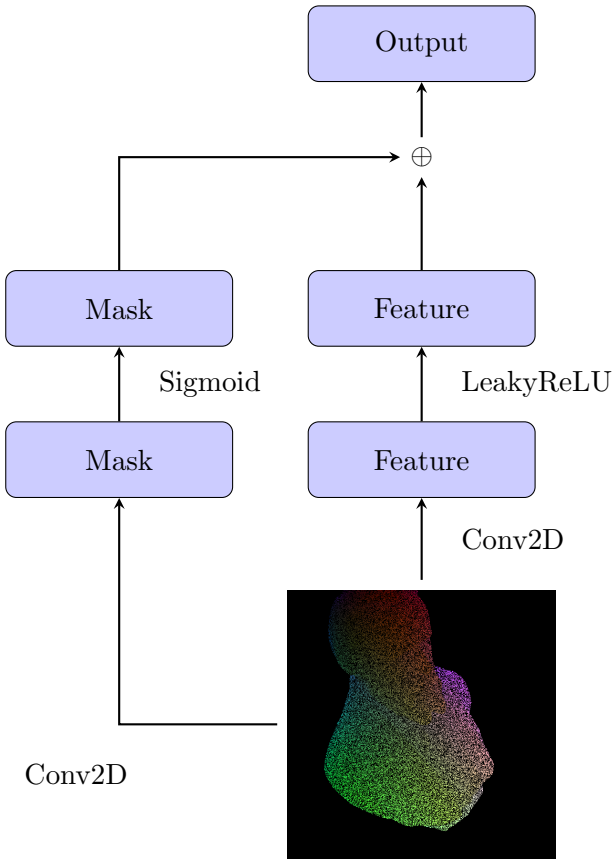


FIGURE 3.1: Gated Convolution Layer, where $\oplus$ denotes element-wise multiplication.

3.2.2 Architecture

Based on the implementation mentioned above, a Gated Convolutional Neural Network based on UNet proposed by Ronneberger, Fischer, and Brox, 2015 is proposed to perform normal inference. The architecture of network is shown in Figure 3.2. It takes 3D vertex with size $512 \times 512 \times 3$ as input, then samples downward 3 times, each scale with 3 gated convolution layers. 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. It the end, two standard convolutional layers have been added, the output normal map has the same size as the input 3D vertex map.

3.2.3 Loss Function

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

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

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.

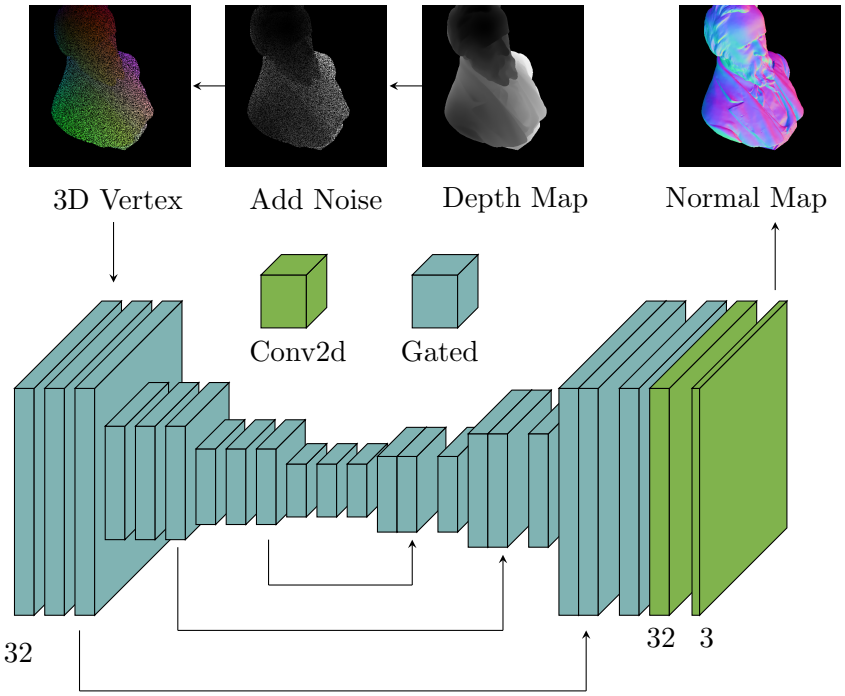


FIGURE 3.2: Basic Normal Neural Network model based on Gated Convolution layer and UNet architecture.

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

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

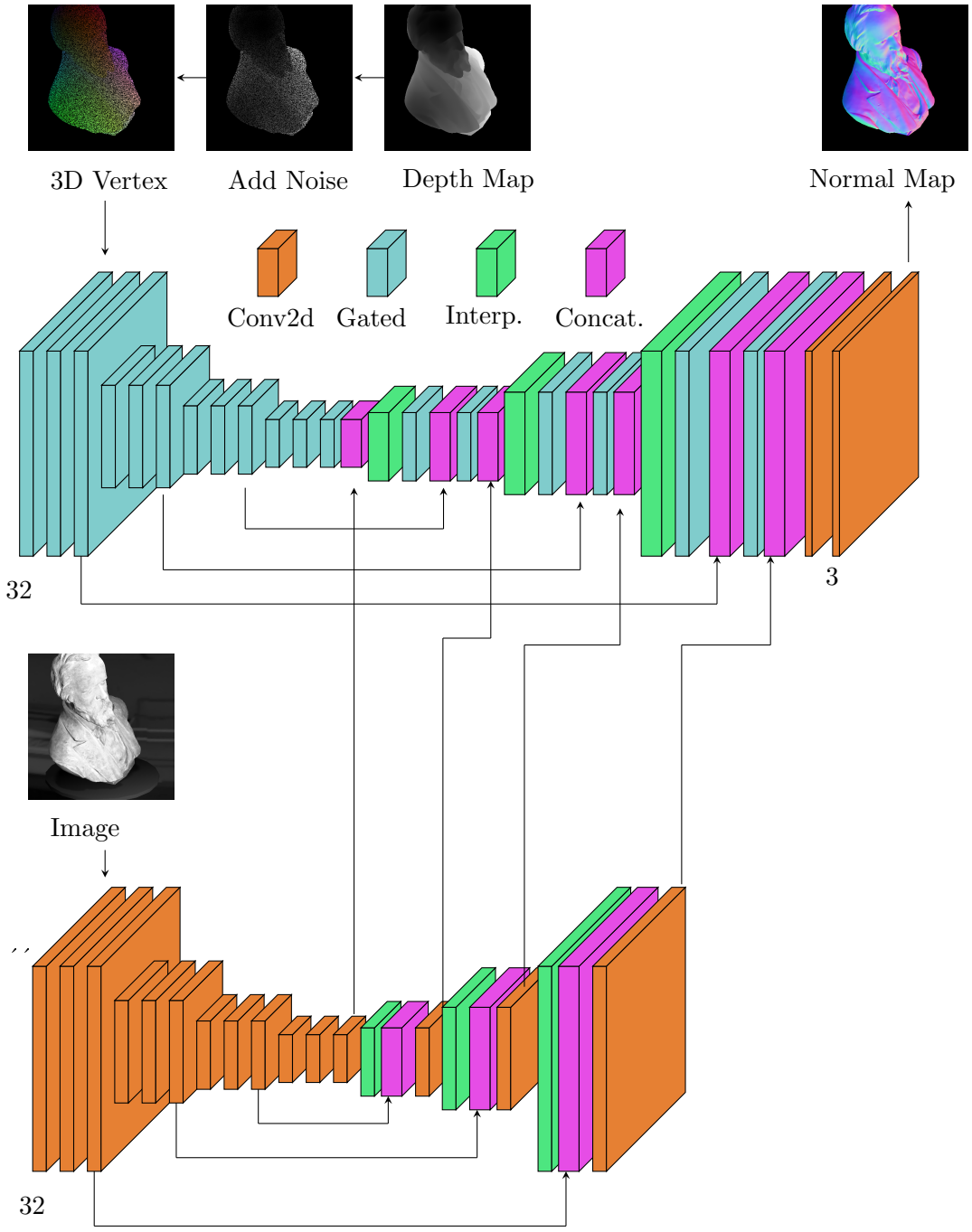


FIGURE 3.3: Guided Gated Convolution Neural Network for normal estimation. The normal branch shows on the upper side taking point cloud as input. The image branch shows on the lower side taking image as input. There are total 4 times fusions between the two branches. The output is a corresponding normal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

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)...

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

Chapter 4

Dataset

4.1 Synthetic Dataset

For each object in the resources, a set of generated synthetic 3D scene via Unity is used for each object. The main advantage using generated scene is the complete information of all the information in the synthetic world. The depth map can be captured in a loss-free way. The corresponding normal map can also be safely considered as ground truth. To construct the dataset, 1000 different scenes have been saved. The model pose, camera position, light position are random changed in each scene. For each scene, following information is recorded

- Depth range
- Light position
- Camera intrinsic and extrinsic matrix
- Depth map
- Grayscale image
- Normal Map
- Grayscale Image

In Unity, the grayscale image in acquired via RGB texture by the following equation:

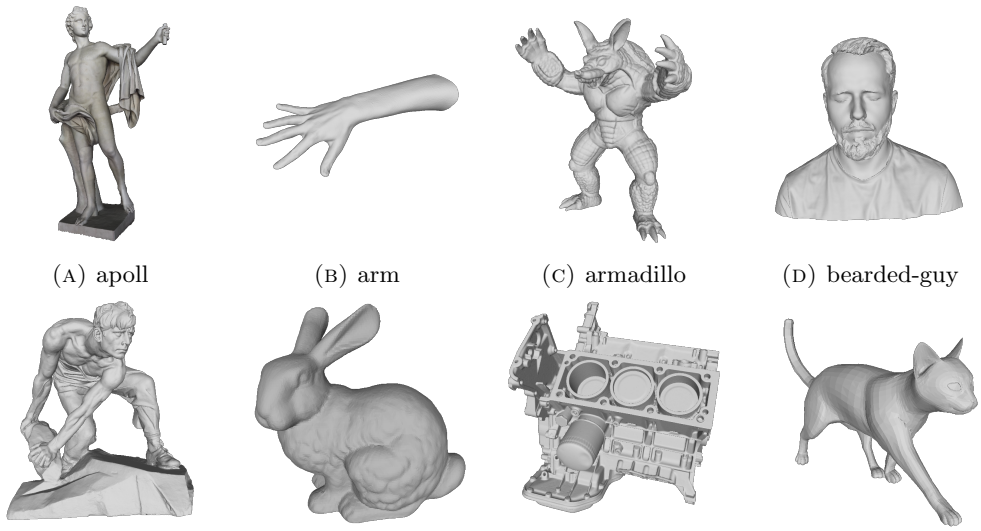
$$gray : \frac{r + 2g + b}{4}$$

4.1.1 Resource

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 powerful sensor is required to perfectly record every detail of the models.

McGuire, 2017, McGuire, n.d. and *Smithsonian 3D Digitization* n.d. published a sets of point cloud dataset with high resolution detail. The model has its point position, normal direction and uv coordinates, which as be used as ground truth in the training work. In this master thesis, 50 models are selected as training dataset, 5 are selected as test dataset. Figure **??** illustrates some of the point clouds. Appendix A gives a full version of dataset models.

FIGURE 4.1: Depth Map of an object captured by Kinect



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

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

FIGURE 4.2: Some point clouds in training dataset

4.1.2 Surface Normal

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).

4.1.3 Point Cloud

A depth map D is captured by a depth camera like Kinect, which is a 1 channel image that contains the information relating to the distance of the surfaces of the scene objects from a viewpoint. The range of distance depends on the performance of depth camera. It can be saved as a 16-bit image, i.e. each pixel in range $0\tilde{6}5535$. A depth map with knowing calibration can be further converted to a structured point cloud.

4.1.4 Point Cloud calculation

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 value $D(u, v)$ is the Z component of the 3D point $P_C = (X, Y, Z)$ corresponding the camera origin. Based on the triangle similarity, we can get other two components X and Y as follows

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

where fk_u, fk_v is the focal length in pixels align u and v axes. It can be further converted to world coordinate system based on extrinsic matrix R and t, that is

$$P_W = P_C R + t$$

4.1.5 Point Cloud Normalization

The vertices have been normalized before feed them into the model. The depth range of each scene is shown in Figure **??**. 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. Figure **??** shows the fluctuation of extreme values and their ranges in 100 random training items. Table 4.1 gives the corresponding average values.

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

TABLE 4.1: The ranges and extreme values of each axis. The extreme min and max values of both X axis and Y axis are close to -0.75 and 0.75 separately. The case for Z axis is 6.5 and 8.0 separately. However, the range of three axes are relatively similar, around 1.5.

For the normalization, first we translate the point to the original point as much as possible, then choose the range value of one axis as a scale factor, normalize it to a unit object.

$$U^a = (V^a - V^a_{min})/s \quad \text{for } a \text{ in } X, Y, Z \text{ axis} \tag{4.1}$$

where V^a_{min} denote the minimum value appeared in axis a, s is the range of an axis, which can be any one of X, Y, or Z axis.

In this thesis, the range of X is chosen as scale factor for each scene.

As discussed in 4.1.5, the 3D vertex point cloud should be moved to origin point and normalize in range $[0, 1]$ to acquire a scale invariant feature.

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

Table 4.2: The structure of a single tensor in the dataset.

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

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.

4.1.6 Noise

The raw depth maps captured by Kinect usually have missing pixels. As shown in Figure **??**. Observing the depth map, the missing pixels distributed all around the scene. Therefore an uniformly distributed pixel-delete noise is used for noise simulation. It is reasonable that the noise intensities in different scenes are various. Some scenes have more missing pixels and some have less. It enables the model to not only learn scenarios with noise, but also with small minority noise or even without noise. In the noised dataset, the noise intensity is a x-percent pixel dropoff operation. For example, noise-intensity-10 means removes 10% pixels. Figure 4.3 shows the visualization of depth map after adding noise.

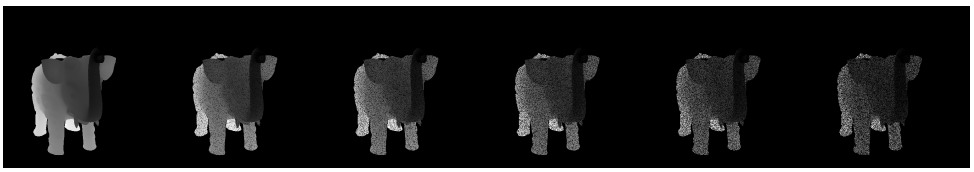


Figure 4.3: Noise-intensity on 0, 10, 20, 30, 40, 50. Object Name: elephant-zun-lid.

4.1.7 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 4.2.

4.2 Real Dataset

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

4.3 Metrics for evaluation

Angle Loss

Chapter 5

Experiments

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

5.1 Gated Convolution Neural Network for Normal Inference

The first neural network that worked well for normal prediction in this master thesis is introduced in this section. It is named Gated Convolution Neural Network, or GCNN. The GCNN model is nothing but a UNet with gated convolution layers. it can predict normal map based on a point cloud. The error is around 10 degrees.

The model uses a 512x512x3 3D vertex matrix as input and the output is a 512x512x3 normal map. The architecture is based on description in 3.2.2. The channel size is up to 32 through the whole network. For the training parameters, learning rate is 0.001, penalty is 1.4, optimizer is Adam optimizer, the loss function is penalty-l2 loss as described in **??**.

The evaluation result on 90 test scenes is shown in Figure 5.1. 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.

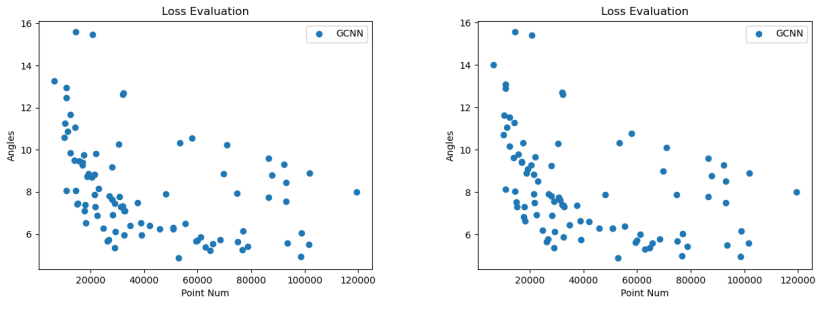
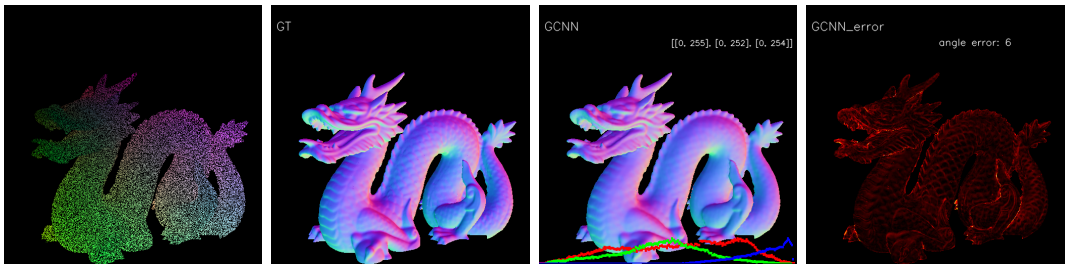


FIGURE 5.1: Evaluation of average angular loss on the whole test dataset with 90 scenes. The x-axis indicates the point number, the y-axis indicates the angles. The **Left** one using point cloud without noise, the **right** one has noise.

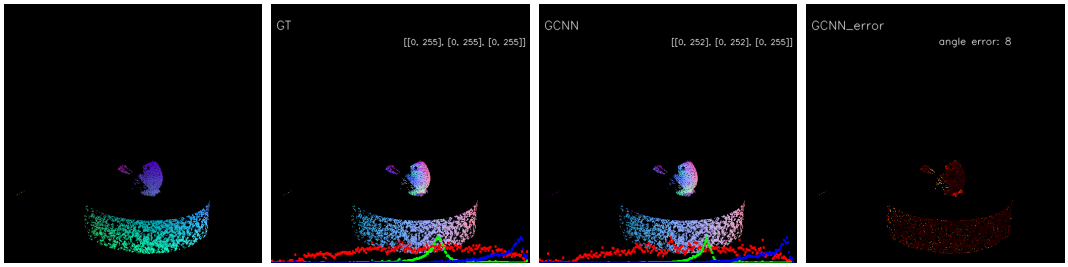
The evaluation visualization on synthetic dataset is shown in Figure 5.2. The error concentrate mainly on severe changed surface region, like the teeth, paw and horn of the dragon.

The evaluation visualization on real dataset is shown in Figure 5.3



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

FIGURE 5.2: GCNN Normal Inference on Synthetic Dataset (object: dragon)



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

FIGURE 5.3: Evaluation on Real Dataset

5.2 Guided Gated Convolution Neural Network for Normal Inference

The GCNN model is nothing but a UNet with gated convolution layers. it can predict normal map based on a point cloud. The error is around 10 degrees. guided-GCNN

5.3 GaRes Model

Figure **??** shows the ground truth and predicted normal map of a Washington statue. In the predicted normal map, as shown on the right side, the relief on the side of stone chair is lack of sharpness compare to the ground truth on the left side.

To further visualize the error of predicted normal map, figure **??** shows the angle error of normal map. It is obvious to see, that the error goes higher in the coarse surface, like fingers, gown and relief. Oppositely, the error goes lower in the smooth surface, like the arm, face, and foot. The coarse surface are mainly the boundaries, or edges, which can be extract efficiently using edge detection algorithms, like Canny Edge detector. Figure **??** shows the detected edge of ground truth using Canny Edge Detector.

Chapter 6

Conclusion

Gated convolution neural network...

Appendix A

Dataset

A.1 Dataset

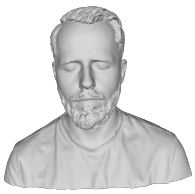
A.2 How do I change the colors of links?

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}`.

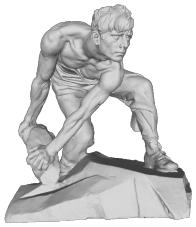
(A) apoll

(B) arm

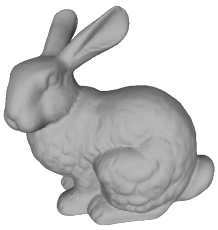
(C) armadillo



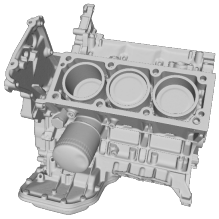
(D) bearded-guy



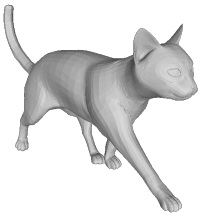
(E) bronze-sculpture



(F) bunny



(G) car-engine



(H) cat

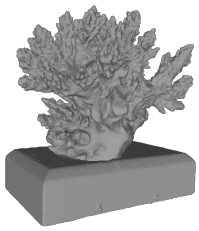
(I) side-chair

(J) chair2

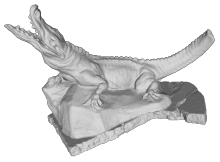
(K) chandelier

(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

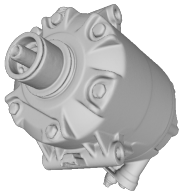
(V) green-hat

(W) happy-budda

(X) heart-pendant

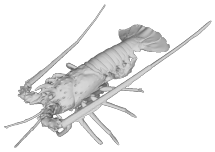
FIGURE A.1: Point clouds in training dataset A

(A) indonesian-statue

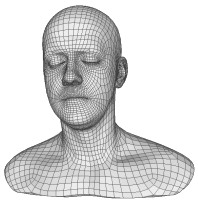


(B) industrial-compressor

(C) kylix



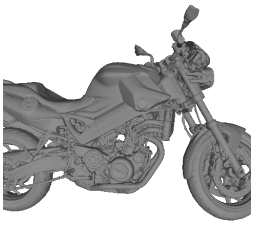
(D) lobster



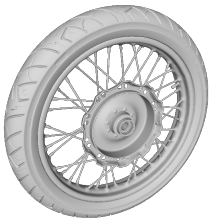
(E) head

(F) Lu-Yu

(G) Maitreya

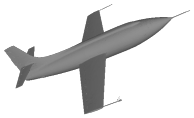


(H) motorbike



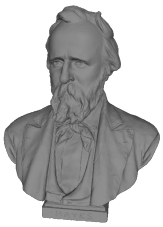
(I) motorcycle-wheel

(J) Pan



(K) plane

(L) elkins-refrigerator

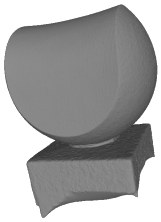


(M) Rutherford

(N) statue-dragonfly-tamer

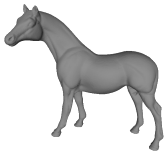
(O) Tiger

(P) transmission



(Q) universe

(R) Vestalin



(S) Zebra

(T) elephant-zun-base

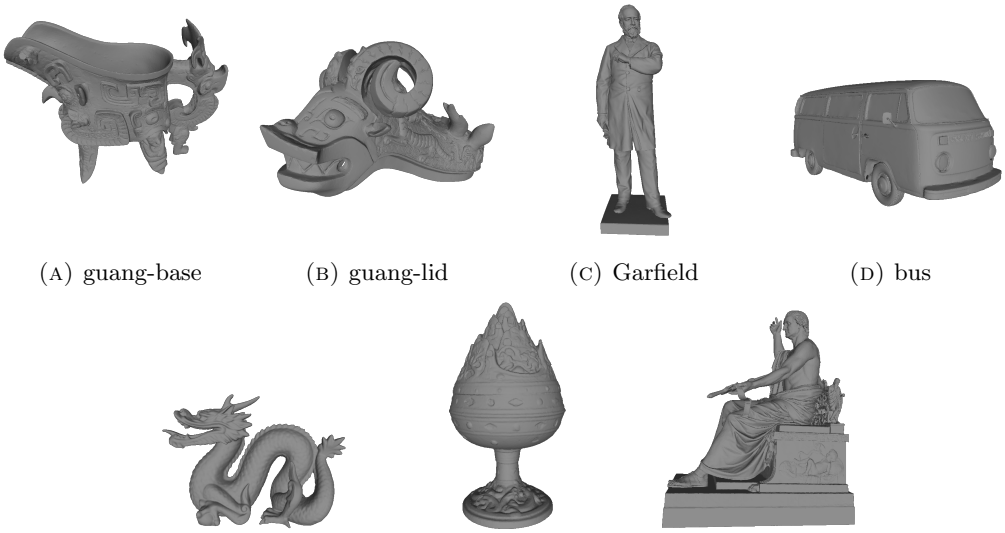
(U) elephant-zun-lid

(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.

Knutsson, Hans and Carl-Fredrik Westin (Jan. 1993). "Normalized and Difierential 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.

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.

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].