The approaches are trained on dataset "synthetic-50-5" as mentioned in Chapter **??** with 3000 scenes. The input matrix has 128×128 height and width. Each vertex map has dimension $128 \times 128 \times 3$, light map has $128 \times 128 \times 3$, image has dimension $128 \times 128 \times 1$. The model is trained with PyTorch 1.10.0a0, CUDA 11.4.1, GPU with single NVIDIA GEFORCE RTX 3090.

1 GCNN model evaluation

The GCNN model is the base model of the whole thesis. In this section, we first evaluate the inpainting performance of the model based on light map input. Then we use the same model for normal inference task.

1.1 Light Inpainting

The light inpainting task in this section in order to evaluate the noise inpainting performance of GCNN model. It takes semi-dense light map as input to predict the fully dense light map, as shown in Figure 1.

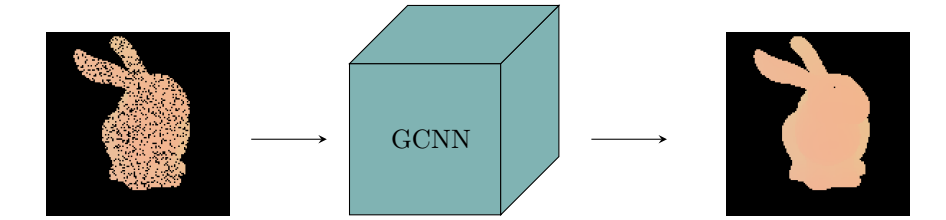


Figure 1: Light map inpainting based on GCNN architecture.

The network is trained on 3000 light maps in the "synthetic-50-5" dataset with initial learning rate 0.001, learning schedule 8,1000 with decay factor 0.5, the batch size is 8, the feature map channel is 128 in all of the gated convolution layers. The result is shown in Table 1 In order to show the performance of the model in a better way, we use two extra model as the comparison. The first one is GCNN-NOC, where NOC means "no skip connection", the second one is CNN, which has the same architecture as the GCNN but only use standard convolution layers in the network.

Model	Angle	Time /ms	bz	lr-schedule	lr-df
GCNN	0.17	4.72	8	8,1000	0.5
GCNN-NOC			8	8,1000	0.5
CNN				8,1000	0.5

Table 1: The inpainting performance of the GCNN model for light map inpainting. The angle error is the average angle error of all valid pixels in the test case. bz stands for batch size, lr-schedule stands for learning rate schedule, lr-df stands for learning rate decay factor.

Figure 2: GCNN Normal Inference for light map inpainting based on dragon object. From left to right, grayscale image, semi-dense light map(input), ground-truth light map, predicted light map, the angle error of the predicted light map.

1.2 Surface Normal Inference based on GCNN

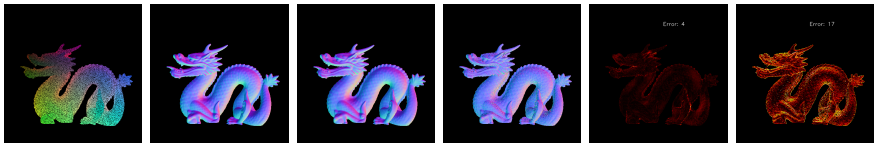
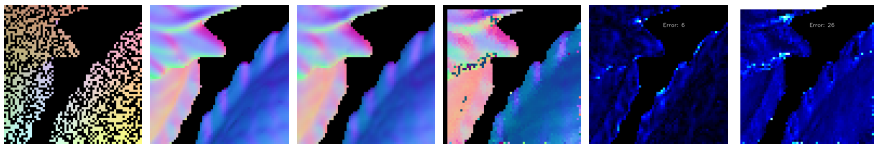


Figure 3: GCNN Normal Inference on Synthetic Dataset (object: dragon) From left to right: Input vertex map, ground-truth, GCNN normal map, SVD normal map, GCNN error map, SVD error map

We evaluate the performance of the normal inference task based on a single GCNN architecture. The model is trained on 3000 images from "synthetic50-5" dataset. The training pipeline use batch size 8, Adam optimizer (**adam**), learning rate of 1×10^{-3}, learning schedule [8,1000], learning decay factor 0.5.

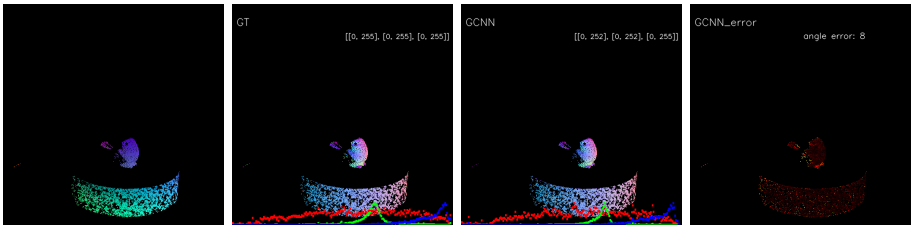
The model takes a single vertex map as input and predict the corresponding normal map in the output. A qualitative evaluation on object "dragon" is shown in Figure 3 and Figure 4, the neighbor based approach based on SVD optimization is considered as a base line. The GCNN based approach has apparently better performance than SVD based approach, which get rid of the trace of the noise in the input vertex map. Figure 4 zooms in the hindhead area of the dragon object which provides a closer comparison with the ground-truth. As shown in the figure, the area of beard of the dragon and its neck are on the different surface, which can be easily seen on GCNN result, whereas the SVD based approach is confused and get a very randomly surface normal.



(a) Input (b) GT (c) GCNN (d) SVD (e) (f)

Figure 4: Zoom in of dragon object evaluation.

The evaluation visualization on real dataset is shown in Figure 5



(a) point cloud (b) ground-truth (c) GCNN (d) Angle Error

Figure 5: Evaluation on Real Dataset

We evaluate the GCNN model on 100 test images as a quantitative evaluation. The result is shown in Figre 6. 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.

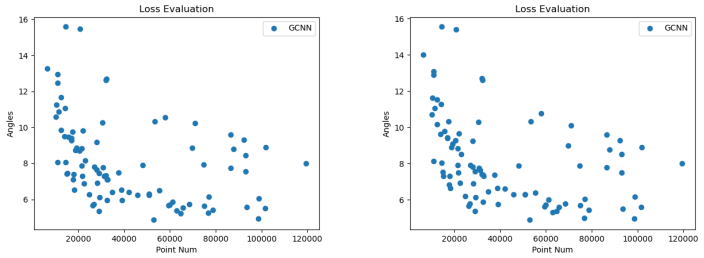


Figure 6: Evaluation of average angular loss on the whole test dataset with 90 scenes. The x-axis indicates the point number, the y-axis indicates the angles. The **Left** one using point cloud without noise, the **right** one has noise.

2 Guided Gated Convolution Neural Network for Normal Inference

2.1 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 7. 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.

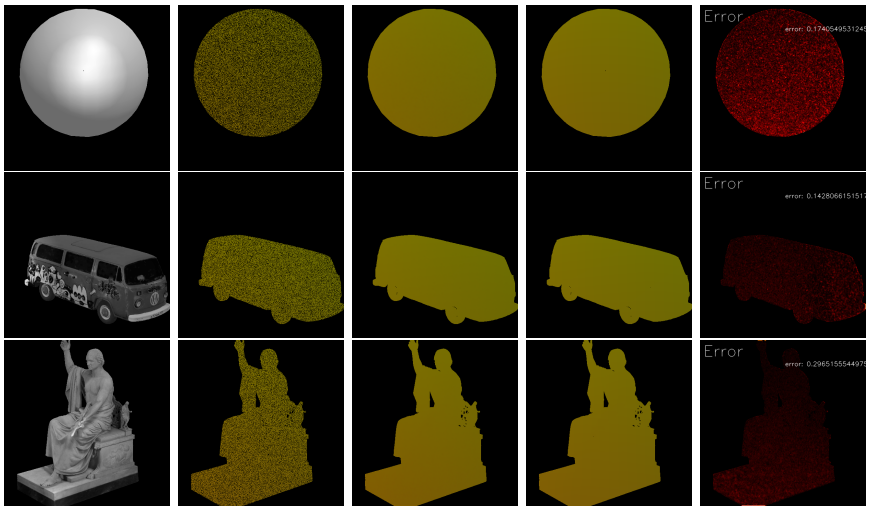


Figure 7: Qualitative evaluation of light net on Sphere, bus and Washington statue. From left to right, grayscale image, semi-dense light map(input), full-dense light map(output), full-dense light map(ground-truth), error map.

Figure 8 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 2. 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.

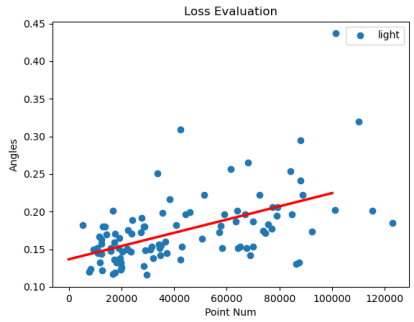


Figure 8: The evaluation on 100 test cases of Light Net.

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

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

2.2.1 title

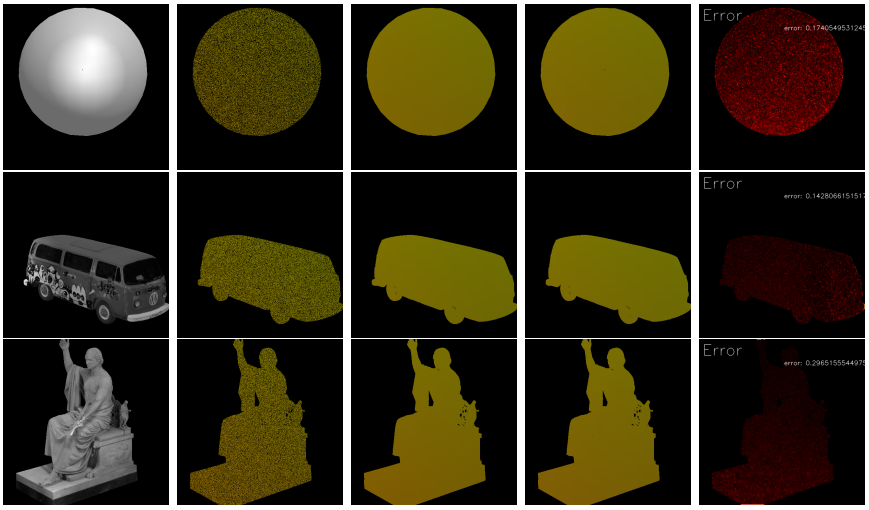


Figure 9: Qualitative evaluation of Trignet net on Sphere, bus and Washington statue. From left to right, semi 3D vertex map, full-dense normal map(output), full-dense normal map(ground-truth), error map.

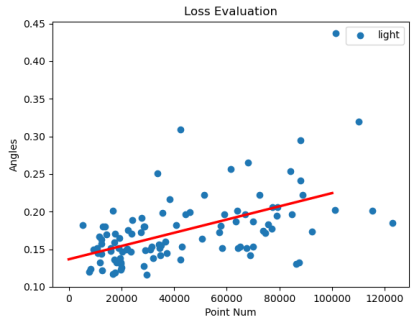


Figure 10: The evaluation on 100 test cases of Trignet.

2.3 comparison

From the Figure 12 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 2: 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

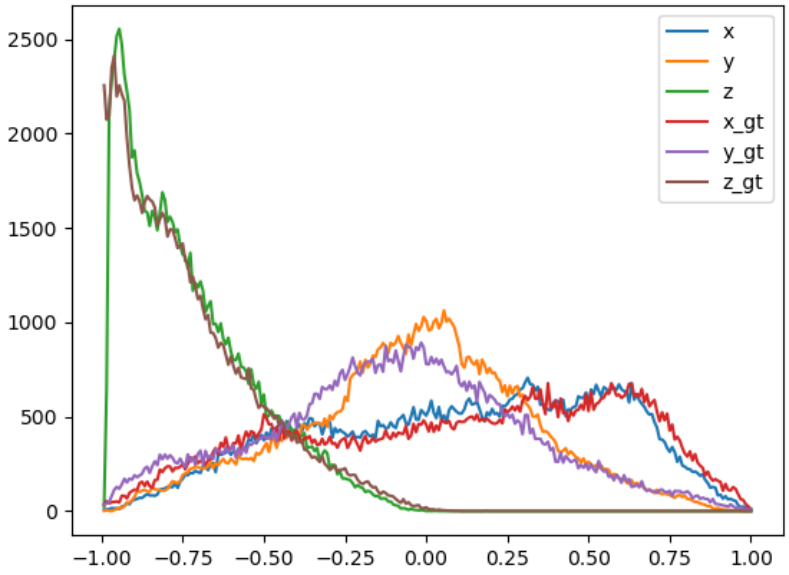


Figure 11: The normal difference of between GCNN and ground-truth in x, y, z-axis respectively. The y axis indicates the number of points, x axis indicates the value of normal in x/y/z axis. (The chart is based on the "dragon" scene showing above)

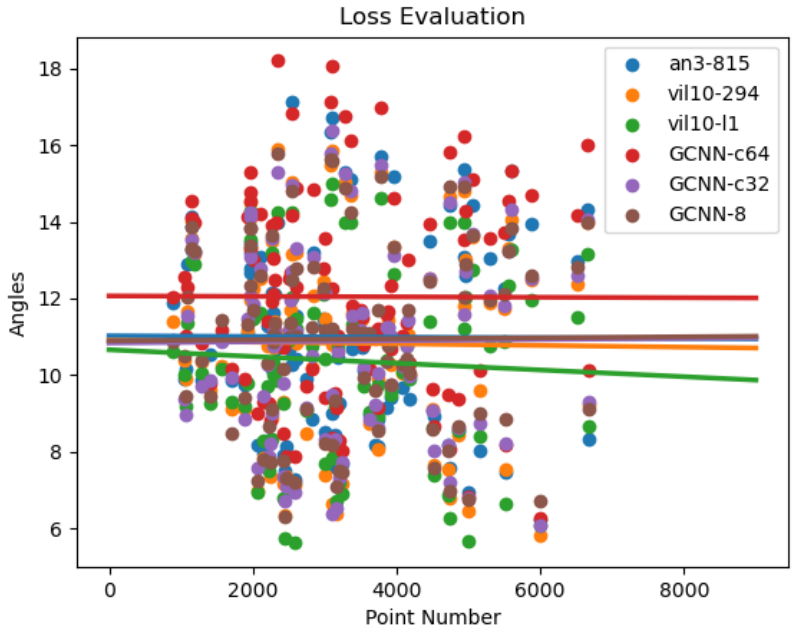


Figure 12: The normal difference of between GCNN and ground-truth in x, y, z-axis respectively. The y axis indicates the number of points, x axis indicates the value of normal in x/y/z axis. (The chart is based on the "dragon" scene showing above)