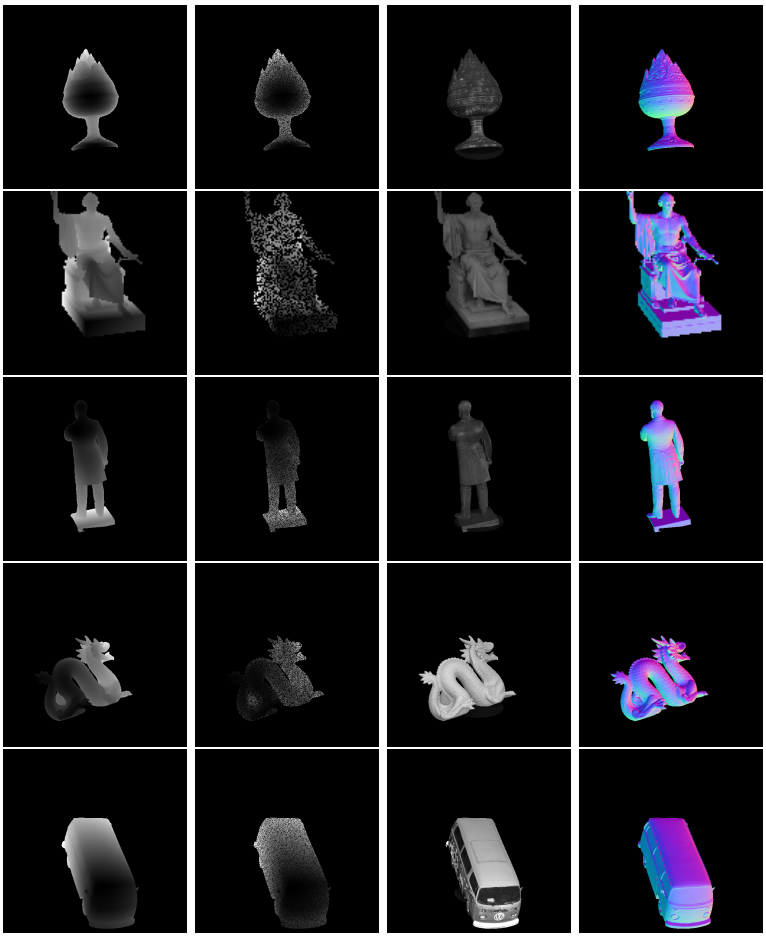
1 Training Details

The approaches are trained on dataset "synthetic-50-5" as mentioned in Chapter **??** with 3000 scenes. Each scene has a depth map with dimension $128 \times 128 \times$ in height, width and channel, an image with dimension $128 \times 128 \times 1$. The depth map is converted to 3D vertex map as introduced in Chapter **??**. The light map is calculated based on vertex map and the known light position. We create a tensor in PyTorch that includes vertex map, image and the light direction for each scene and considered it as one training case. Thus 3000 scenes has corresponding 3000 training cases. Each scene has a corresponding ground-truth normal map for loss calculation and the evaluation.



(a) Depth Map (b) Add noise (c) Image (d) Normal

Figure 1: Some of the test scenes during the training. From top to bottom, baoshanlu, Washington, Garfield, Dragon, Bus

The training processes are evaluated in every epochs with 29 evaluation scenes that model never seen before, which contains the 5 different objects in the "synthetic-50-5" test set. Figure 1 shows some of the test scenes during the training work. Note that the position of objects are not placed naturally on the stage but with a random rotation in X, Y, Z axes, respectively.

The training pipeline use batch size 8, Adam optimizer (**adam**), learning rate start from 1×10^{-3}, learning schedule [8,1000], learning decay factor 0.5. The model is trained with PyTorch 1.10.0a0, CUDA 11.4.1, GPU with single NVIDIA GEFORCE RTX 3090. It takes 14 hours to train GCNN and 35 hours to train the Trip-Net. We terminate the training when the evaluation on the test dataset converged.

2 GCNN model based on Geometry Information

The GCNN model is the base model of the whole thesis. The architecture is described in **??**. We use a single GCNN to estimate the surface normal based on geometry information. It uses vertex map as input to estimate the corresponding tangent surface normal map.

In order to verify the applicability of UNet architecture and Gated Convolution layer for the normal inference task, two similar models are created. We replace all of the gated layer to standard convolution layers in the network but keeps all of the other settings same in model "CNN". It is used to verify the performance the gated convolution layers. As mentioned in chapter **??**, the gated layer is designed to deal with noised input. Since all of the vertex map in the dataset has been added noise, the GCNN is supposed to over-perform "CNN". Another model called "NOC" is designed to verify the skip connection in the UNet, which simply removes the skip connections in the network but keeps other settings same. Is is designed to show the validation of skip connections. Figure 2 shows the training history on BerHu Loss. The GCNN approach achieves a lower loss from start to the end of the training.

3 Trip-Net model based on Calibrated Illuminated RGB-D Image

The Trip-Net model uses three times GCNN architecture with 4 times fusions, which is more difficult to train. It takes the calibrated illuminated RGB-D images as input to estimate the surface normal map. For the sake of comparison, we take the GCNN model as a baseline, to observe the beneficial of illuminated information using with Trip-Net architecture. Since it is more complicate than GCNN, we also explored the optimum fusion times of the Trip-Net to see any possibility for the model simplification. A set of similar models have been trained with same settings but different fusion times, denotes by Trip-Net-FxF,

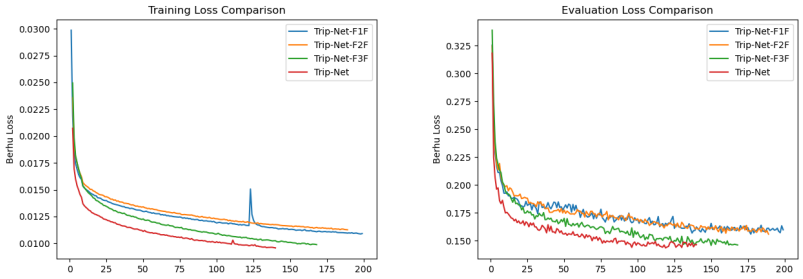


Figure 2: The training history of GCNN model. The line chart records the training BerHu loss of the model GCNN, NOC and CNN. The left shows the training loss history whereas the right one shows the evaluation loss history.

where x denotes the fusion times. We evaluate the fusion times from 1 to 4. For the learning rate. we set $1e-3$. It goes well with GCNN model but lead to loss explosion in Trip-Net. Thus we set a learning rate schedule with an extra decay step at epoch 8. The decay factor is 0.5. The batch size is chosen as 8.

Figure 3 shows the training history of these models on BerHu Loss. As shown in the loss figure, all of the four models has a reasonable learning rate. Trip-Net with four times fusion converges faster than others and also achieve a lower loss. Trip-Net-F3F converges slower than Trip-Net but achieved to a similar evaluation loss. The evaluation loss in Trip-Net-F1F and Trip-Net-F2F are relative higher than 3 or 4 times fusions model.

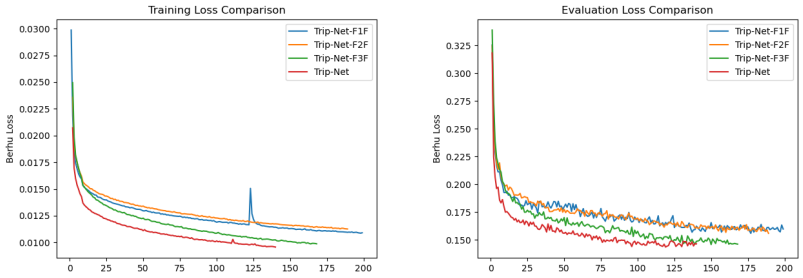


Figure 3: The training history of Trip-Net model. The line chart records the training BerHu loss of the model Trip-Net, Trip-Net-F1, Trip-Net-F2, Trip-Net-F3, GCNN.

However, the sacrifice of accuracy gives a relatively lighter model. Since we remove the fusions between different pipes, the corresponding upsampling layers in the image and light pipes can also be removed. The model can be trained faster and the size is reduced as well. Table **??** gives a comparison of the size and training time among different models.

Model	#Total	V-P	L-P	I-P	Size /MB	Time /h
Trip-Net-F1F	88	40	24	24	106	9.82
Trip-Net-F2F	92	40	26	26	137	7.35
Trip-Net-F3F	96	40	28	28	167	7.35
Trip-Net	100	40	30	30	198	9.15
GCNN	32	32	0	0	46	2.18

Table 1: Model information. Columns V-P, L-P and I-P represent the number of convolution layers in vertex pipe, light pipe and image pipe respectively. Note that one gated convolution layer is constructed with 2 standard layers, thus it is counted as 2.

4 Evaluation

We evaluate the trained models on "synthetic-50-5". 5 objects are considered in the test dataset. They are: $Baoshanlu$, Bus, $Dragon$, $Garfield$, and $Washington$. Each object has 20 scenes with total 100 scenes for 5 objects. The test objects do not exist in the training dataset. We evaluate all the presented models on the test dataset, in order to fit them in one table, the name of each models are simplified. SVD model use SVD optimization method, NOC model is the no skip connection version of $GCNN$, CNN is the CNN version of $GCNN$. $F1$, $F2$, $F3$,$F4$ means the fusion times in the Trip-Net.

Among all the columns, the most important models are $GCNN$ (depth map based) and $F4$ (calibrated illuminated RGB-D image based), which is the core result of this thesis. When evaluate the $GCNN$ models, we can take SVD model as baseline, NOC and CNN are used to verify the performance of $GCNN$ model. When evaluate the $F1-F4$ models, we can take $GCNN$ model as baseline.

Based on metrics proposed by **geometry'based'solution**, 6 different metrics are used for evaluation. Note that the input vertex map is only semi-dense. One of the benefit of GCNN architecture is the robust to the noisy input, thus in the evaluation, all the points including missing points in the input vertex map are taken into account.

Average Angle Error Metric The metric calculate the average angle error for each point between the inferred normal and ground-truth normal.

Median Angle Error Metric The metric calculate the median angle error of all the point in the normal map.

5 Degree Error Metric The metric calculate the percentage of the predicted normals that has error less than 5 degrees comparing to ground-truth.

11.5 Degree Error Metric The metric calculate the percentage of the predicted normals that has error less than 11.5 degrees comparing to ground-truth.

22.5 Degree Error Metric The metric calculate the percentage of the predicted normals that has error less than 22.5 degrees comparing to ground-truth.

30 Degree Error Metric The metric calculate the percentage of the predicted normals that has error less than 30 degrees comparing to ground-truth.

Object	#	**SVD**	**GCNN**	**NOC**	**CNN**	**F1**	**F2**	**F3**	**F4**
Baoshanlu	20	35.66	11.09	13.58	15.55				9.82
Bus	20	31.93	7.79	8.95	11.93				7.32
Dragon	20	39.57	10.60	15.29	16.03				7.35
Garfield	20	39.69	10.20	12.50	14.46				9.15
Washington	20	42.83	13.43	17.59	18.71				12.13

Table 2: Average Angle Error of the evaluation dataset.

Object	#	**SVD**	**GCNN**	**NOC**	**CNN**	**F1**	**F2**	**F3**	**F4**
Baoshanlu	20	34.06	8.86	10.82	13.25				7.62
Bus	20	34.14	4.44	5.02	8.69				4.01
Dragon	20	36.43	7.62	11.10	13.26				5.06
Garfield	20	37.60	6.40	8.90	11.31				5.81
Washington	20	36.89	7.64	11.38	13.64				6.76

Table 3: Median Angle Error of the evaluation dataset.

Object	#	**SVD**	**GCNN**	**NOC**	**CNN**	**F1**	**F2**	**F3**	**F4**
Baoshanlu	20	0.01	0.25	0.18	0.11				0.31
Bus	20	0.00	0.56	0.50	0.23				0.59
Dragon	20	0.00	0.31	0.17	0.10				0.50
Garfield	20	0.00	0.41	0.27	0.14				0.45
Washington	20	0.00	0.38	0.26	0.10				0.41

Table 4: Percent of error less than 5 degree of the evaluation dataset.

Object	#	**SVD**	**GCNN**	**NOC**	**CNN**	**F1**	**F2**	**F3**	**F4**
Baoshanlu	20	0.03	0.62	0.52	0.41				0.69
Bus	20	0.05	0.81	0.78	0.65				0.83
Dragon	20	0.02	0.69	0.51	0.40				0.82
Garfield	20	0.03	0.72	0.62	0.51				0.75
Washington	20	0.02	0.62	0.50	0.40				0.66

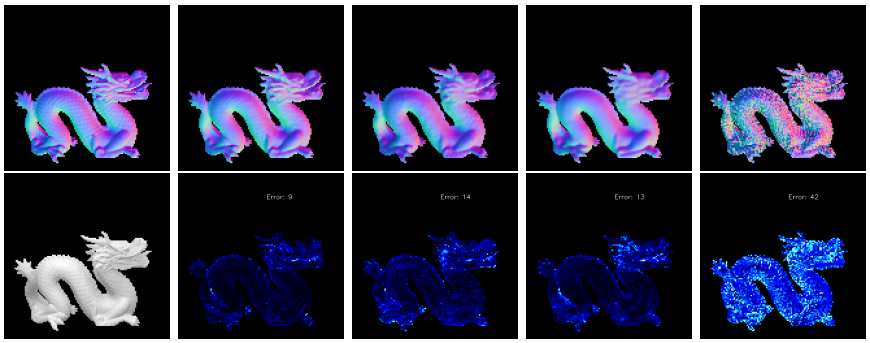
Table 5: Percent of error less than 11.5 degree of the evaluation dataset.

Object	#	**SVD**	**GCNN**	**NOC**	**CNN**	**F1**	**F2**	**F3**	**F4**
Baoshanlu	20	0.18	0.90	0.84	0.79				0.92
Bus	20	0.26	0.93	0.91	0.89				0.94
Dragon	20	0.14	0.90	0.79	0.80				0.95
Garfield	20	0.13	0.89	0.86	0.84				0.91
Washington	20	0.14	0.81	0.72	0.72				0.84

Table 6: Percent of error less than 22.5 degree of the evaluation dataset.

Object	#	**SVD**	**GCNN**	**NOC**	**CNN**	**F1**	**F2**	**F3**	**F4**
Baoshanlu	20	0.37	0.96	0.93	0.90				0.97
Bus	20	0.43	0.96	0.94	0.93				0.96
Dragon	20	0.30	0.95	0.88	0.90				0.98
Garfield	20	0.27	0.94	0.92	0.91				0.95
Washington	20	0.28	0.88	0.81	0.82				0.90

Table 7: Percent of error less than 30 degree of the evaluation dataset.

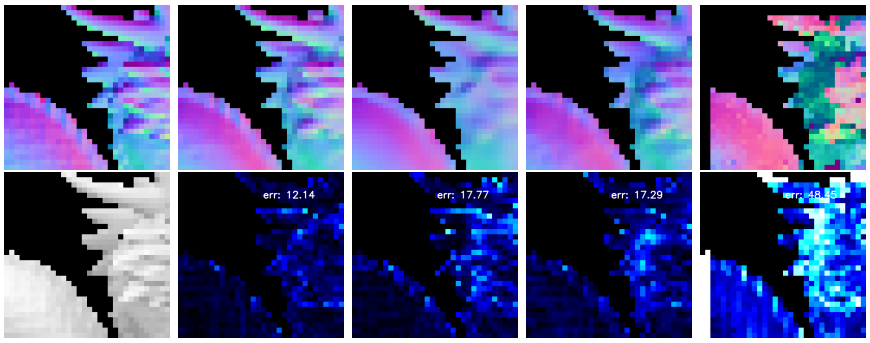


(a) GT (b) GCNN (c) CNN (d) NOC (e) SVD

Error: 0 90

Figure 4: Surface Normal Inference based GCNN model on "Dragon´´ object. The first row shows the estimated surface normal. The second row is the angle error map.

A qualitative evaluation on object "dragon" is shown in Figure 4. SVD approach is considered as the baseline shown in last column. As shown in the figure, learning based methods performs better than SVD in terms of angle error. The SVD approach is failed to deal with semi-dense input since there exists many points that missing neighbors. The GCNN model is especially good at noise input due to the gated convolution layer design. As a further detailed comparison, 5 gives a closer visualization on the same object.



(a) GT (b) GCNN (c) NOC (d) CNN (e) SVD

Figure 5: Zoom in of the center region of Dragon object. The first row is surface normal, the second row is the corresponding errors. NOC model has no skip connnection, CNN model replace gated convolution layer to standard convolution layer.

As shown in figure 5, the GCNN method gives a sharper edge prediction on the horn area of the dragon object, as well as the scales, whereas the no skip

version (NOC) is blurry in the same area.

The CNN version has the skip connection thus gives a better detail than NOC model. However, if we compare the error map of GCNN and CNN in figure 4, the CNN has less accurate in the smooth area than GCNN model. Like the dragon body, CNN model has a overall higher error than GCNN. It is because the noise of the input still disturb the CNN model and it takes the input noise into account for normal estimation which deviate to the correct surface normal. When we look back to the GCNN based method, we can found that the surface normal has better performance in the smooth area compare to the CNN approach and a sharp detail compare to the no skip connection version.

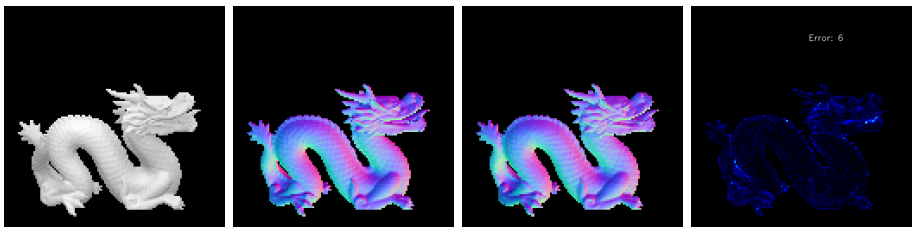
Table 8 gives a quantitative evaluation for GCNN model. It bases on 100 different test scenes in the "synthetic-50-5´´ dataset with angle metrics for evaluation.

Model	**Angle**	**Time /ms**	**bz**	**lr-schedule**	**lr-df**
SVD(baseline)	41.14	320.40	8	8,1000	0.5
GCNN	10.64	10.44	8	8,1000	0.5
GCNN-NOC	13.61	5.38	8	8,1000	0.5
CNN	15.35	4.15	8	8,1000	0.5

Table 8: The performance of the GCNN model for geometry information based normal inference. 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.

5 Surface Normal Inference based on Calibrated Illuminated RGBD images

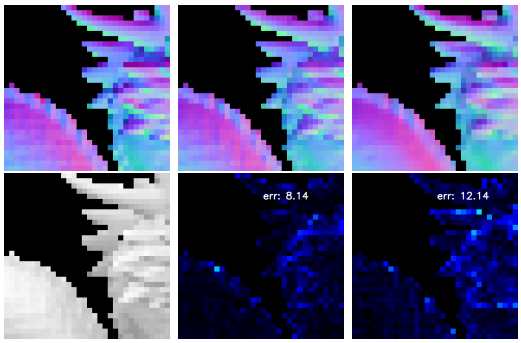
For the approach using illuminated calibrated RGBD image, the task is undertaken by Trip-Net introduced in **??**. The qualitative evaluation is shown in figure 6. As a comparison, we placed GCNN result in the last column. The training settings for all the models are exact the same to ensure fairness. As shown in the figure, TripNet uses illuminated calibrated RGBD image has a better performance than GCNN model. The dragon scales are sharper in TripNet result. Figure 9 gives a closer visualization.



(a) Image (b) GT (c) Predict (d) Error

Error: 0 90

Figure 6: Normal inference based on Trip-Net. Test image resolution 128×128



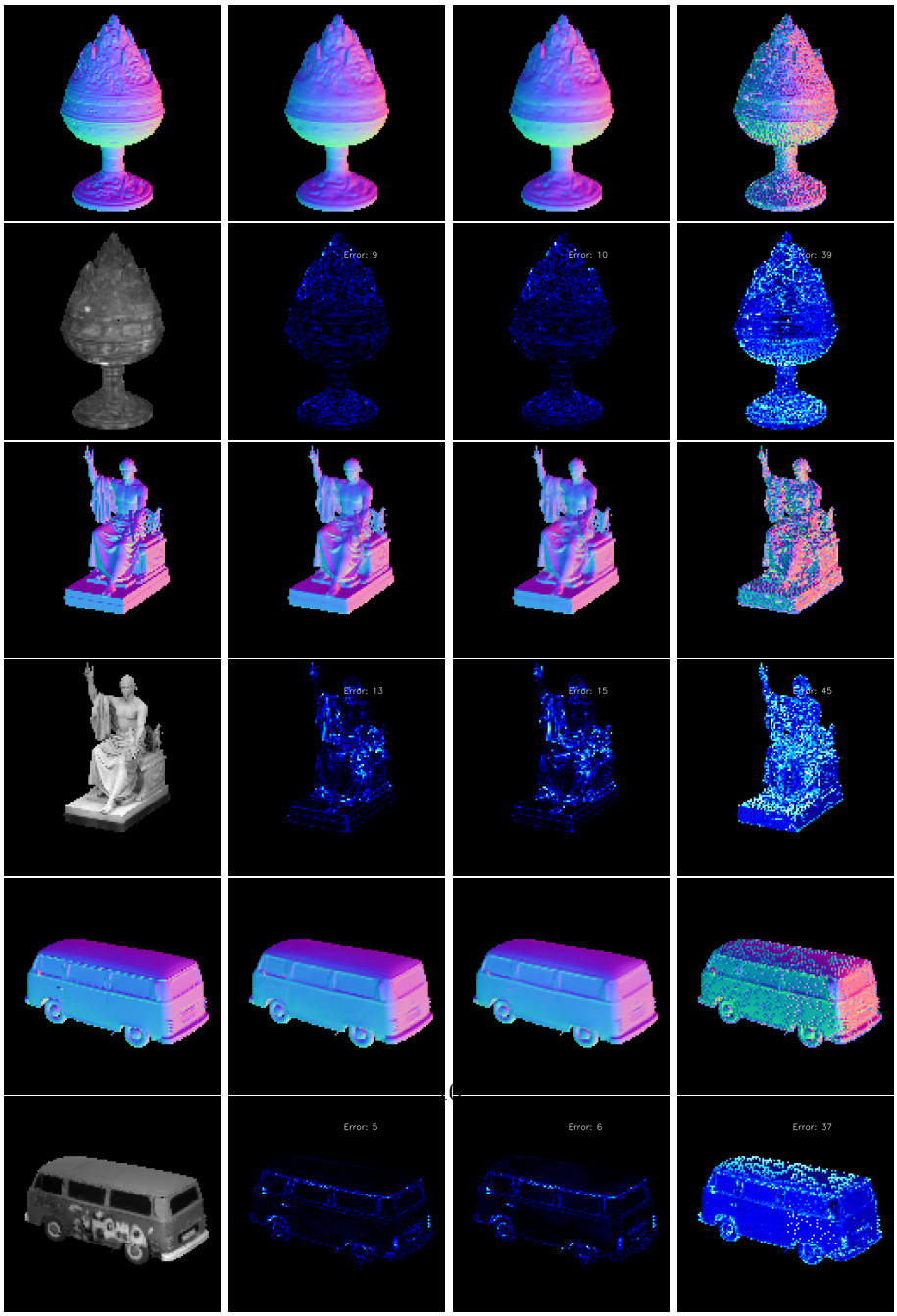
(a) GT (b) Trip-Net (c) GCNN

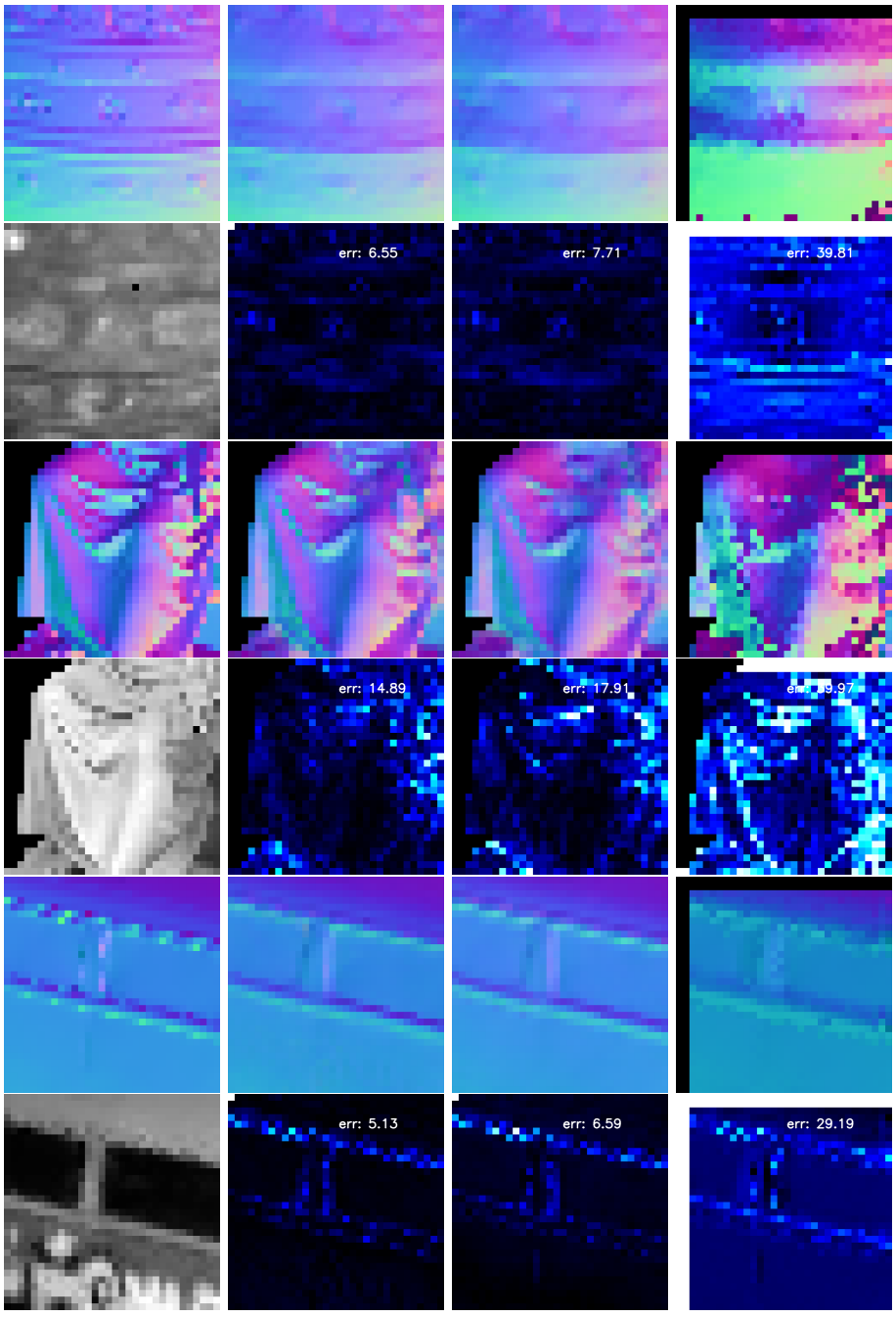
Figure 7: Zoom in of the center region of Dragon object. The first row is surface normal, the second row is the corresponding errors.

5.1 Comparison

Model	Angle	Time /ms	bz	lr-schedule	lr-df	l/i. Nr.
SVD	41.14	320.40	-	-	-	0
GCNN	10.64	10.44	8	8,1000	0.5	0
TripNet-CNN	10.46	28.74	8	8,1000	0.5	1
TripNet-F1B	9.22	44.59	8	8,1000	0.5	1
TripNet	**9.17**	43.79	8	8,1000	0.5	1

Table 9: A quantitative evaluation on proposed approaches. The angle error is the average angle error of all valid pixels in the test case. The time unit is in 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





(a) GT (b) Trip-Net (c) GCNN (d) SVD

Figure 9: Zoom in of the center region of the objects in Figure 8

From the Figure **??** 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?

5.2 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 10. 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. All the models are tested on a laptop with a single NVIDIA GeForce 940MX, 2GB memory.

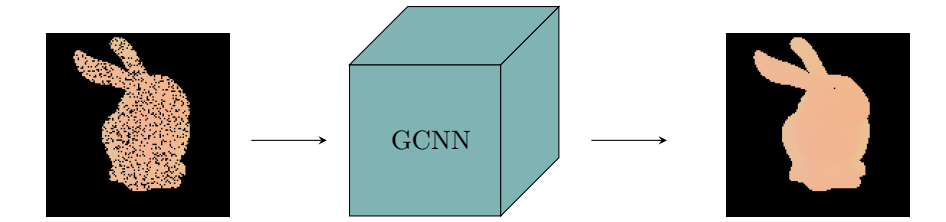


Figure 10: Light map inpainting based on GCNN architecture.

The result is shown in Table 10. Besides the GCNN model, we use two extra model as the comparisons. 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. The GCNN architecture achieves the average angle error lower than 1 degree. As a comparison, U-CNN model has 4 degrees error, which shows the relative weakness of noise filter competence. Besides, the skip connection also booster the performance, since the no skip connection version still has error greater than 1 degree. The both two models for the inpainting task reveals the superiority of GCNN architecture. Figure 12 and 11 gives a quantitative and qualitative evaluation.

Model	Angle	Time /ms	bz	lr-schedule	lr-df
U-GCNN	0.59	14.45	8	8,1000	0.5
U-GCNN-NOC	1.31	16.14	8	8,1000	0.5
U-CNN	4.10	10.52	8	8,1000	0.5

Table 10: The 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.

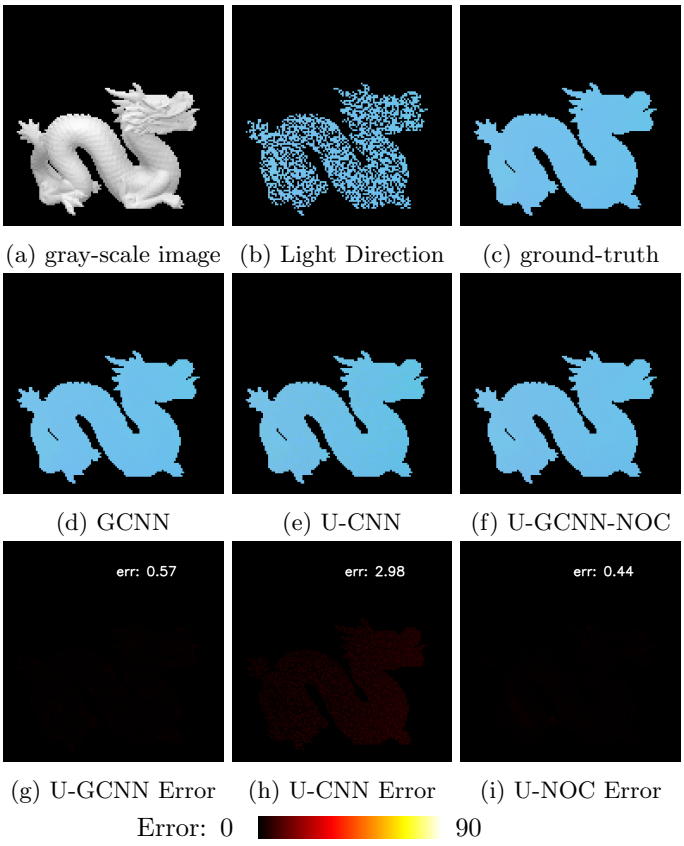


Figure 11: GCNN Normal Inference for light map inpainting based on dragon object. The degree error map is shown in last row. The average error is shown in the upper right corner in the error map.

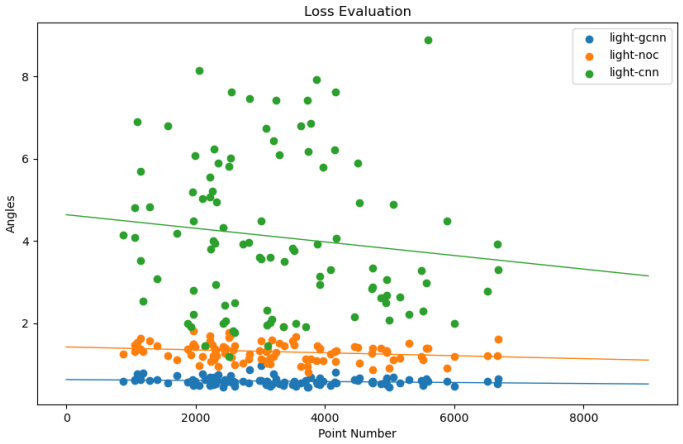
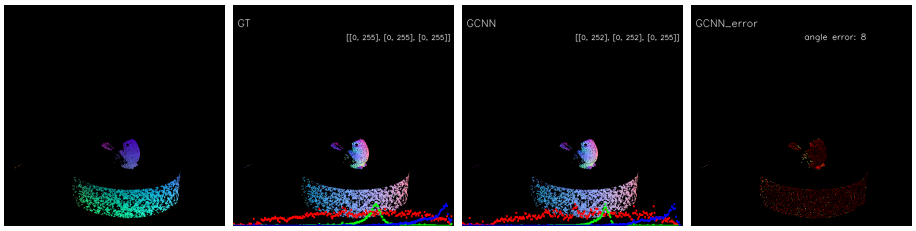


Figure 12: GCNN for light map inpainting on 100 scenes. The evaluated models are corresponding to the table 10, the line in the figure is the corresponding regression line of each model.

The evaluation visualization on real dataset is shown in Figure 13



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

Figure 13: Evaluation on Real Dataset

We evaluate the U-GCNN model on 100 test images as a quantitative evaluation. The result is shown in Figure 14. The GCNN based method has average angle error around 10 degrees, whereas the no skip connection version has error above 12 degrees and the CNN version has error above 15 degrees.

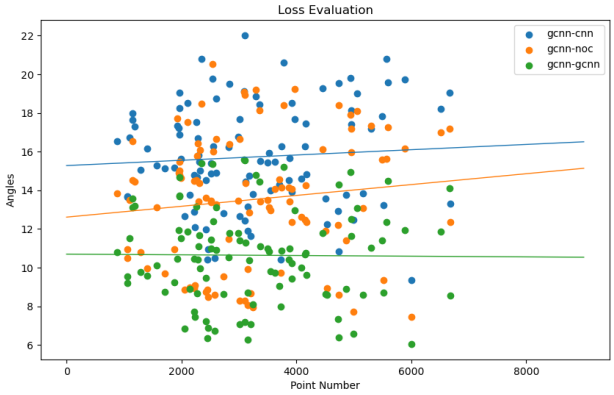


Figure 14: Evaluation of average angular loss on the whole test dataset with 100 scenes. The x-axis indicates the point number, the y-axis indicates the angles. The lines show in the picture are the regression line of each model.