1 Guided normal inference using GCNN

1.1 Intrinsic image decomposition

Intrinsic image model was introduced by **intrinsic-image**. It interprets that an image I can be decomposed as the element-wise product between the reflectance R of the object and the shading S produced by the interaction between light and objects.

$$I = R \odot S$$

It interprets the observed image into reflectance image and the shading image. As shown in Figure 1

Figure 1: Intrinsic image analysis of the bus object. From left to right, original image, reflectance image, shading image, light image, normal image

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.

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

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

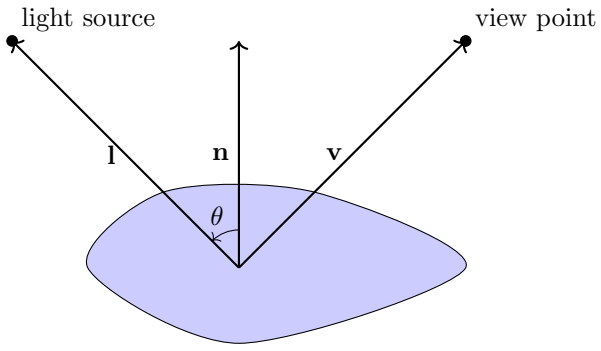


Figure 2: The surface normal, source light direction and the view point direction, where θ denotes the angle between light direction and the normal.

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 images and light directions.

1.2 Light Map

The light map L can be derived from vertex map and the light source position. As shown in Figure 2, the incoming light direction is a vector point from light source to the surface point, therefor it can be calculated as follows

$$L(x,y) \quad = \frac{V(x,y) - (s_x, s_y)}{\|V(x,y) - (s_x, s_y)\|_2} \tag{1}$$

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. The light map is a matrix corresponding the light direction of whole image where each pixel corresponding with each other, thus it has the same size as the vertex map.

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.

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.

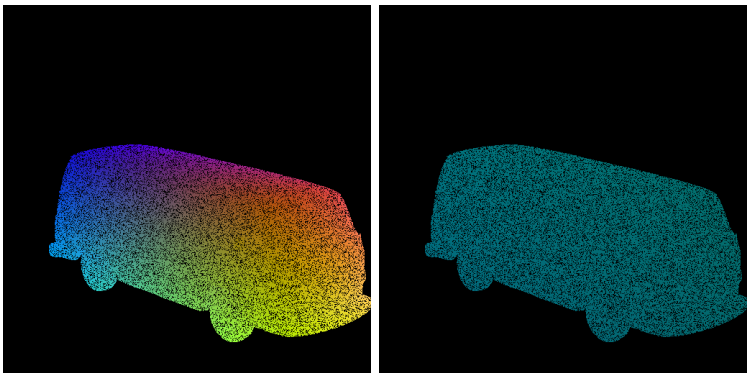


Figure 3: The light map calculated from vertex map and the light source

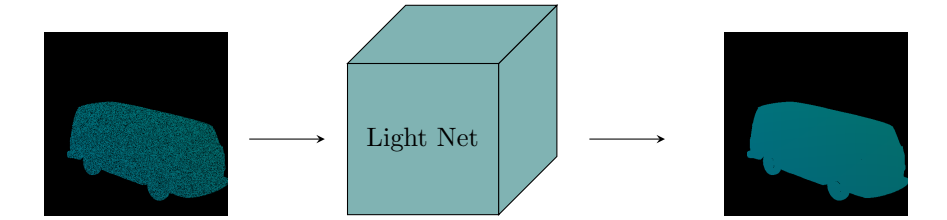
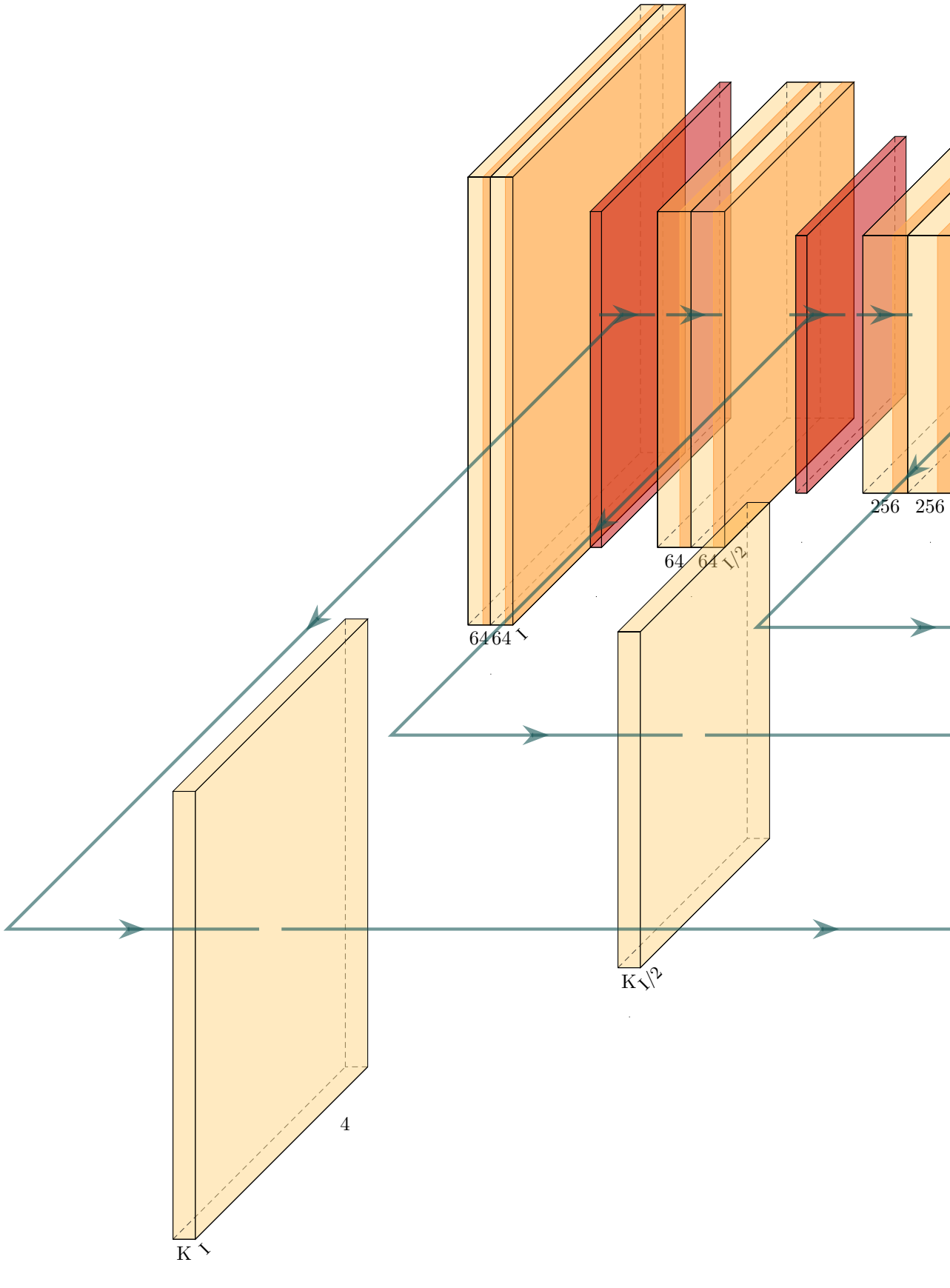


Figure 4: Light Net for light inpainting based on GCNN architecture.

1.3 Light and image guided normal inference using GCNN

Based on above implementations, we propose a Three branches Gated Convolutional Neural Network (TriGNet) to perform guided normal inference. The structure is shown in Figure **??**. As shown in the name, the network utilizes the GCNN architecture three times to accomplish the task.


64
64
1
64
64
1/2
256
256
K
1
4
K
1/2