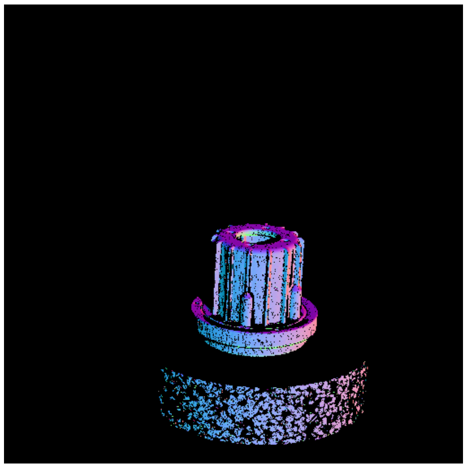
Jingyuan Sha

May 8, 2022

Abstract

Chapter 1

Introduction



Chapter 2

Related Work

In order to estimate normals of an object surface.

In 2012, Holzer et al. [2] presented a read-time method, 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 speed is accelerated via integral image. The drawbacks are, as mentioned in the paper, the normals error go up when point depths change severely.

In 2019, Ben-Shabat et al. [1] presented a CNN based method.

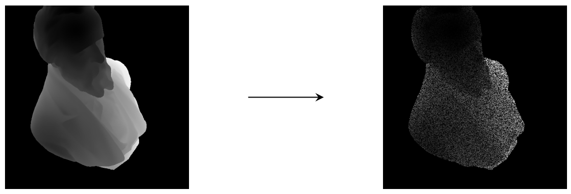
In 2021, Zhou et al. [4]

Chapter 3

Approach

3.1 Dataset

3.1.1 Noise Adding



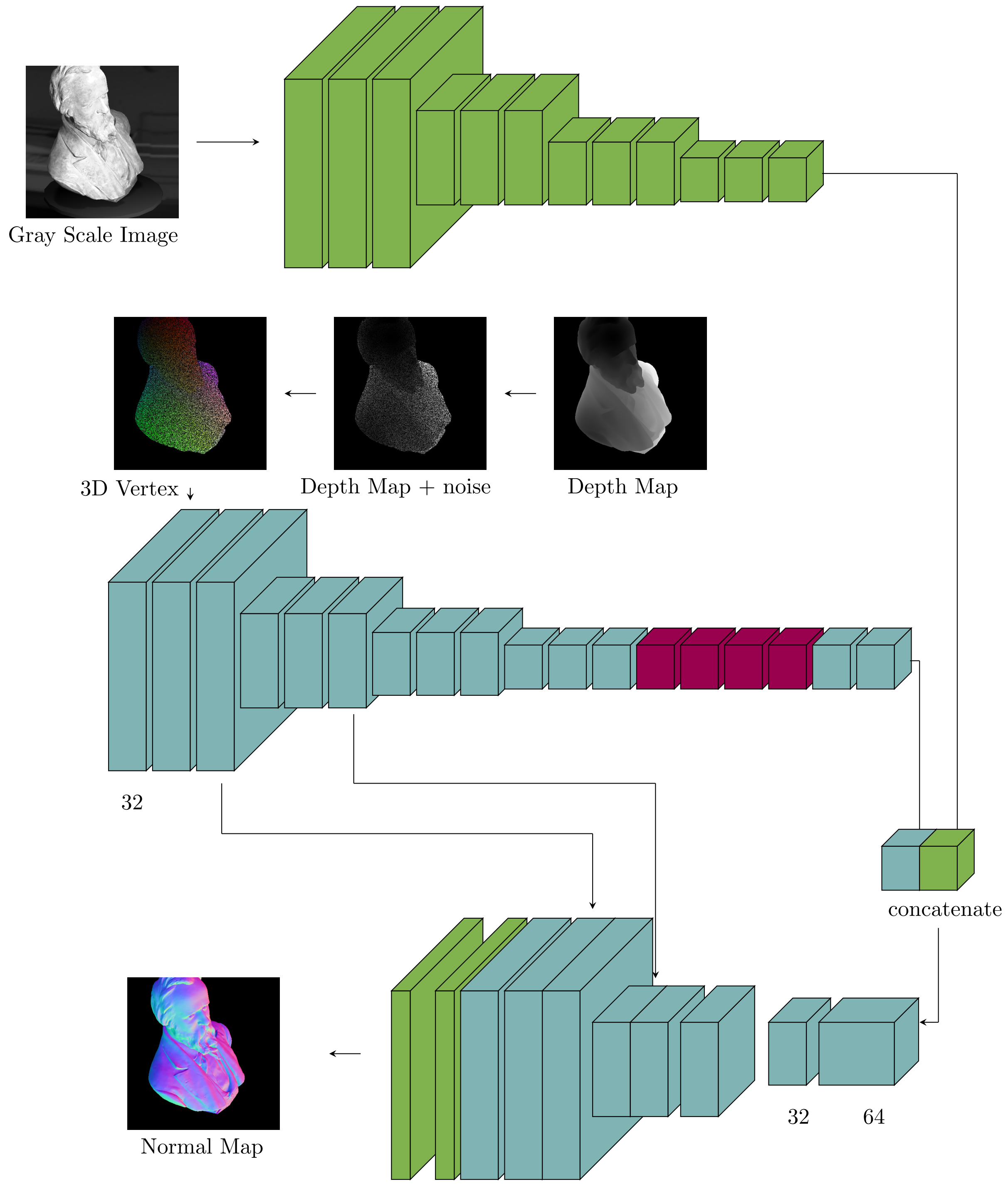
3.2 Gated Convolution

Gated Convolution layer[3], the output of the layer with input size (N, C_{in}, H, W) and output $(N, C_{out}, H_{out}, W_{out})$ can be described as:

$$o(N_i, C_{o_j}) = \sigma\left(\sum_{k=0}^{C_{in}-1} w_g(C_{o_j}, k) \star i(N_i, k) + b_g(C_{o_j})\right) * \phi\left(\sum_{k=0}^{C_{in}-1} w_f(C_{o_j}, k) \star i(N_i, k) + b_f(C_{o_j})\right) \tag{3.1}$$

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 Architecture



Chapter 4

Experiments

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

Chapter 5

Formular

RGB image will be stored as gray value Scene using following equation:

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

5.0.1 Normal from k neighbors

Given a point p locating on plane Π, calculate the normal n of plane Π.

First, find the nearest k neighbors $p_1, p_2, ..., p_k$ of point p using KNN-algorithms. The plane Π containing point p can be fitted using the neighbors of point p. Then the normal is available immediately.

Assume all the neighbors of point p are in plane $\Pi = ax + by + cz + d = 0$. Since we only need calculate the normal, thus with out loss of generation, we can set displacement $d = 0$. Then the normal $\mathbf{n} = (a, b, c)^T$.

Since all the neighbors of point p are located on plane Π, thus we have

$$P_{k\times 3} \cdot \mathbf{n}_{3\times 1} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

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, which can be modeled as following optimization problem

$$\begin{aligned} \min \quad & \|P\mathbf{n}\|^2 \\ \text{s.t.} \quad & \|\mathbf{n}\|^2 = 1 \end{aligned} \tag{5.1}$$

Let the decomposition of $P = U\Sigma V^T$, The solution i.e. normal is the last column of V.

5.0.2 Normalized Convolution

Bibliography

[1] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 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)*, June 2019.

[2] S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli, and N. Navab. 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*, pages 2684–2689, 2012.

[3] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas Huang. Free-form image inpainting with gated convolution, 2018.

[4] Jun Zhou, Wei Jin, Mingjie Wang, Xiuping Liu, Zhiyang Li, and Zhaobin Liu. Fast and accurate normal estimation for point cloud via patch stitching, 2021.