CS276-Assignment 2: Non-line-of-sight imaging

Qihe Chen 2020533088

November 25, 2022

1 Introduction

In this assignment, we implement the filter backprojection (FBP) algorith fo NLOS imaging. The total work is divided into two parts. In the first part, we read the input data and construct the scene with bounding box. In the second part, we implement backprojection for the input data onto the constructed scene and use Laplace filter or threshold as post processing after calculating the heat map.

2 Part 1: Input Data

The time complexity of our backprojection algorithm is $O(N^4)$. Thus, we have to do the downsampling to the input data. Althought it can speed up our algorithm, we lose the accuracy of the input data.

```
cv2.resize(data, None, fx=downsample_factor, fy=
    downsample_factor, interpolation=cv2.INTER_AREA))
```

The interpolation method we choose is cv2.INTER_AREA. We resample using pixel area relation. It may be a preferred method for image decimation, as it gives moire'-free results. But when the image is zoomed, it is similar to the INTER_NEAREST method. And we show the downsample results below.

2.1 Brief explaination for input data

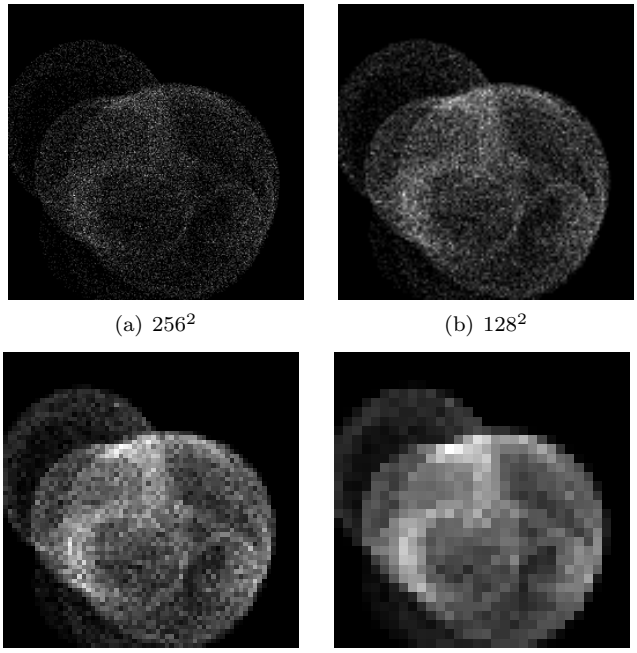
rect_data : 3D matrix, we convert it to the form: Dim_0 is time, Dim_{12} is image.

width : the physical size of wall

bin_resolution and c : are used to compute the radius for backprojection.

2.2 Construct bounding box for scene

We define the wall plane to be $x - z$ plane and the y axis is vertical to the wall plane.



(a) 256^2 (b) 128^2

(c) 64^2 (d) 32^2

Figure 1: Downsample input data comparasions

Then we talk about the range for three axis. Given the confocal NLOS setting, the backprojection region for any point at time t on the wall is a hemisphere with radius r, where

$$r(t) = \text{bin\_resolution} * t * c/2, \qquad t \in [0, 1023]$$

Then the enough range for $x-z$ axis is $[0, \text{width}+2r(t_{max})]$. And the enough range for y axis is $[0, r(t_{max})]$

And the backprojection result without filtering or threshold is below (we just show the result along y axis):

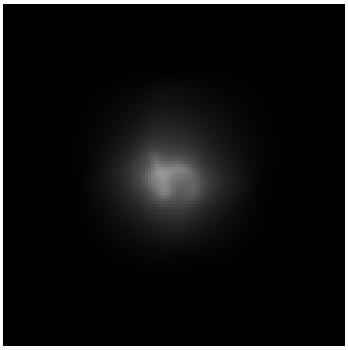


Figure 2: Backprojection result: full bounding box (resolution=32^2)

However, we notice that almost the voxels of our full bounding box have very small densities which can be omitted. Then, we can only consider the region of wall for xz plane, also 25/34 length of y axis since the NLOS image becomes zeros at large time t.

Then we saves more than 10 times memory to save our model. The slice bounding box can give the backprojection result below

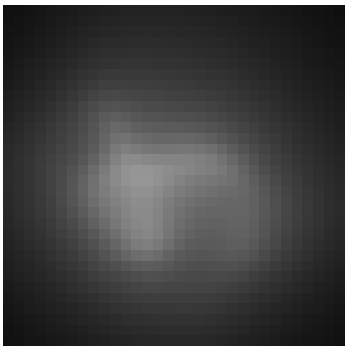


Figure 3: Backprojection result: slice bounding box (resolution=32^2)

3 Part 2: Algorithm

In this assignment, we consider the confocal NLOS setting.

3.1 Voxels and spheres intersection detect

Given a pixel value at position o and time t, we first try to compute $\text{sign}(\|p_v - o\|_2 - r(t))$ for all voxels of bounding box. However, it cannot fully justify whether the sphere determined by $S(o, r(t))$ passes through each voxel.

Therefore, we calculate the distance with eight corners of each voxel to the sphere's central point. $\text{Sum}(v) = \sum_{i=1}^{8} \text{sign}(\|p_{v_j} - o\|_2 - r(t))$. In this way, a voxel passed by sphere if and only if $\text{abs}(\text{Sum}(v)) \neq 8$.

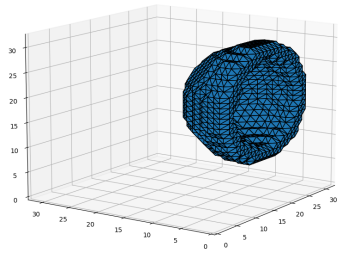


Figure 4: Intersection detect: Backprojection result with single NLOS image

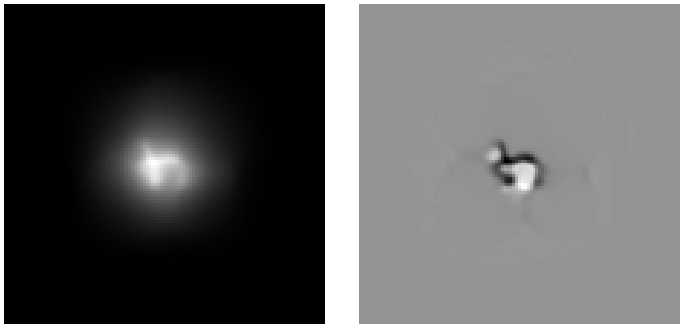
3.2 Filtering

Here, we implement the filtering effect based on the second derivative w.r.t y axis (the direction vertical to the wall). The second derivative is calculated by discretization method.

$$\sigma^*(x, y, z) = \sigma(x, y - 1, z) + \sigma(x, y + 1, z) - 2\sigma(x, y, z)$$

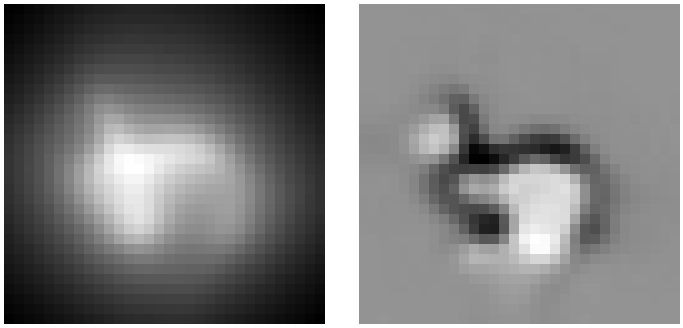
which gives the filtering results what we want. Also, we see the effect of thresholding before applying the filtering, and we find that the thresholding doesn't matter, thus we only use filtering without thresholding as our final post processing.

Here, we show one pair comparsion for different resolution inputdata and with filtering or not.



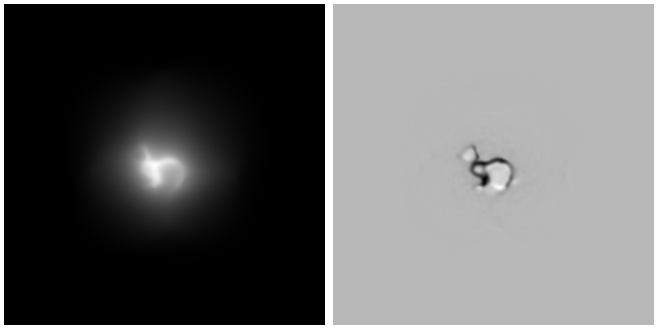
(a) without filtering (b) with filtering

Figure 5: input data resolution=32^2, fullaabb, at $y = 16$



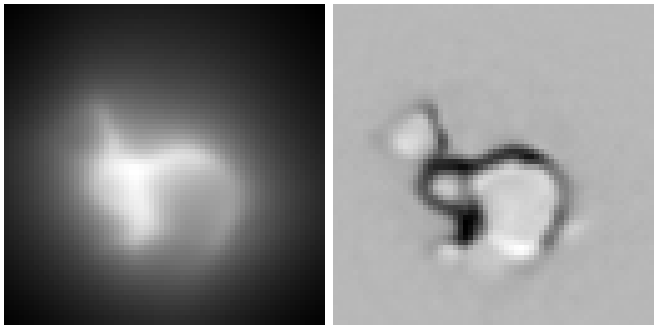
(a) without filtering (b) with filtering

Figure 6: input data resolution=32^2, sliceaabb, at $y = 16$



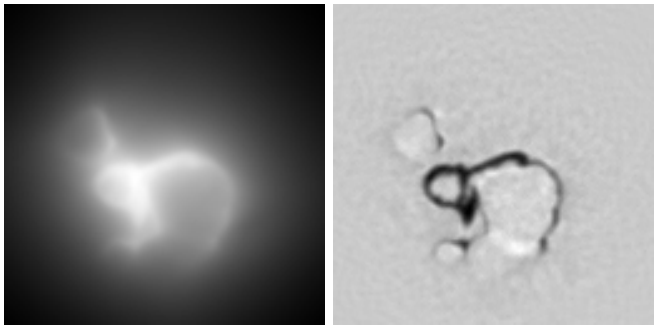
(a) without filtering (b) with filtering

Figure 7: input data resolution=64^2, fullaabb, at $y = 28$



(a) without filtering (b) with filtering

Figure 8: input data resolution=64^2, sliceaabb, at $y = 28$



(a) without filtering (b) with filtering

Figure 9: input data resolution=128^2, sliceaabb, at $y = 52$

4 Analysis

In this assignment, we are able to recover the body of the bunny. However, we can not recover the ears of the bunny. We assume there are three main reasons for why we can not recover the ears of the bunny. The first reason is that the resolution of input data is not enough. The second reason is that the position of the laser is too few. The third reason is that the bunny has strong self-occlusion.

5 Results

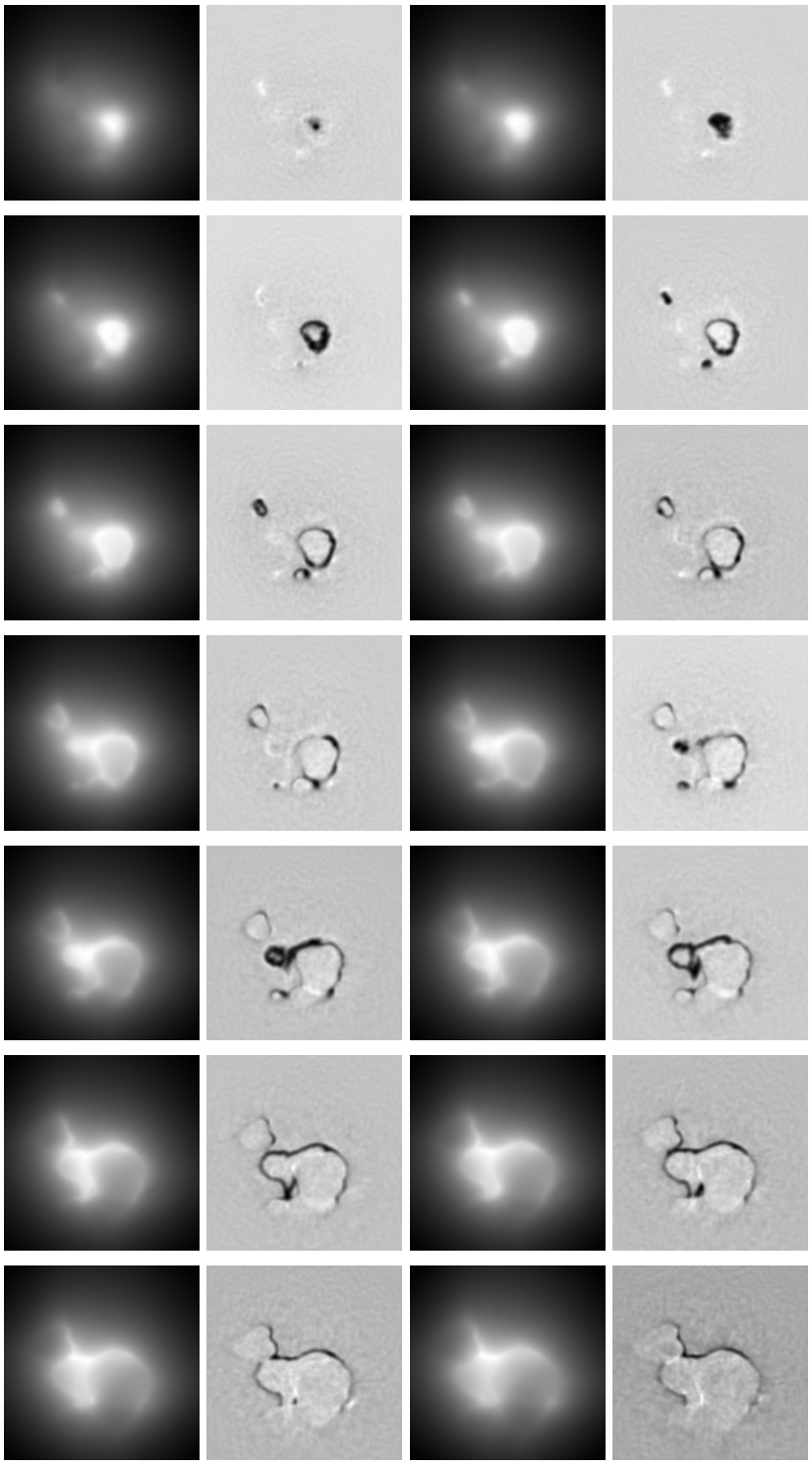


Figure 10: Results: input data resolution=128^2, sliceaabb, $y \in [43, 56]$