Assignment 4: Non-line-of-sight imaging - Lab sessions 7 and 8 -

1 Introduction

The focus of this assignment is implementing a method to reconstruct non-line-of-sight (NLOS) geometry based on time-resolved measurements acquired at a relay wall. Your main task will be developing a NLOS reconstruction algorithm based on back-projection, testing it on different NLOS datasets, and to analyze its performance and ability to reconstruct hidden scenes under different geometric configurations and capture configurations (see Figure 1). In this assignment:

1. You will implement a *naïve* version of the back-projection NLOS reconstruction method described by Velten et al. [2]. Section 2 in this document contains a brief overview of the principal aspects of the method. Please refer to the original article and the lecture notes for more details on the method.

2. You will adapt your method to handle confocal measurements and compare the performance with respect to non-confocal measurements.

3. You will modify your method to mitigate part of the attenuation effects produced by distance falloff and cosine foreshortening and illustrate it in two scenes.

Note that you are not required to provide a high-performance implementation of the back-projection method. It's ok to nest multiple `for` loops as long as the method provides the correct reconstruction.

2 Back-projection reconstruction

In this first task, you will implement a back-projection method to reconstruct NLOS geometry using the transient datasets provided in the supplemental material. You have to implement the basic algorithm introduced by Velten et al. [2] to reconstruct a hidden scene from an arbitrary number of laser and single-photon avalanche detector (SPAD) points on a relay wall. In particular, for every voxel centered at $\mathbf{x}_v$ on a hidden scene, the back-projection reconstruction $G(\mathbf{x}_v)$ of that voxel from a set of laser-SPAD measurements H on the relay wall is defined as:

$$G(\mathbf{x}_v) = \sum_{\forall \mathbf{x}_l, \mathbf{x}_s} H(\mathbf{x}_l, \mathbf{x}_s, t_v) \quad (1)$$

where t_v corresponds to the time-of-flight of third-bounce light paths from the laser device to the relay wall positions $\mathbf{x}_l \in L$, to the reconstructed voxel $\mathbf{x}_v \in V$, back to SPAD positions on the wall $\mathbf{x}_s \in S$, and back to the SPAD device. You can find a schematic of these light paths in Figure 2.

2.1 Datasets

The function `load_hdf5_dataset` reads, processes, and loads the datasets contained in HDF5 files into a MATLAB `struct`, which contains all the necessary information to implement the back-projection algorithm. You can find the description of the output `struct` fields in the comments

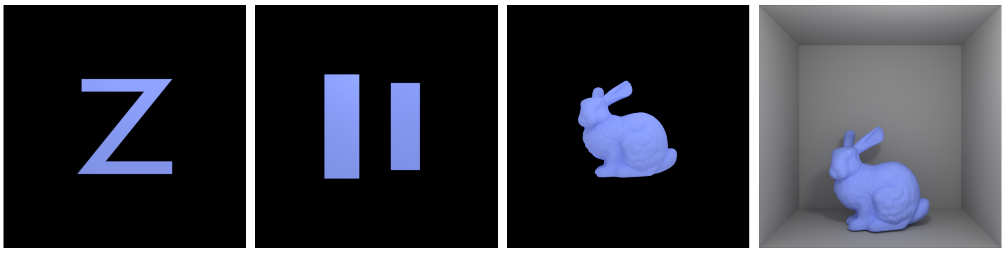

Figure 1: Illustration of the hidden scenes in the four provided datasets, from left to right: `Z_*`, `planes_*`, `bunny_*`, `bunnybox_*`.

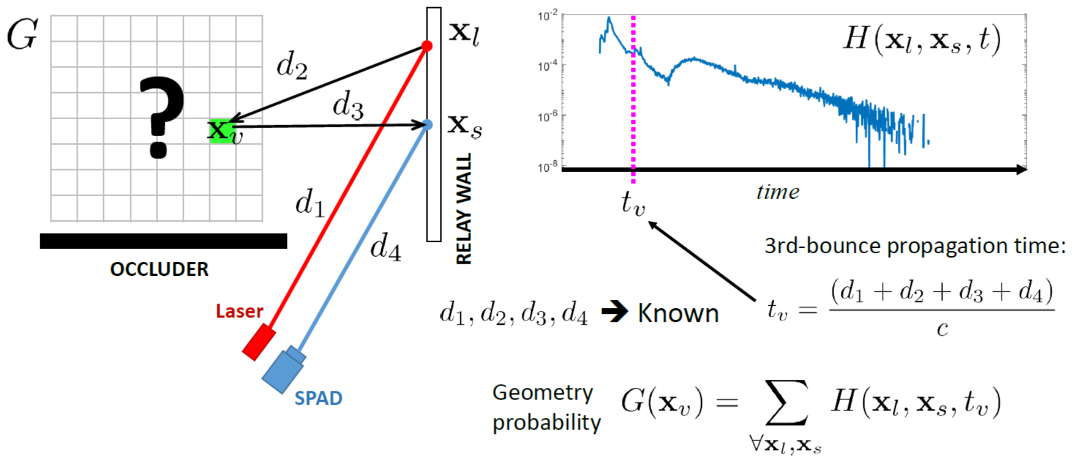
G
?
x_v
d_2
d_3
d_1
d_4
x_l
x_s
RELAY WALL
OCCLUDER
Laser
SPAD
$H(x_l, x_s, t)$
time
t_v
3rd-bounce propagation time:
d_1, d_2, d_3, d_4 → Known
$t_v = \frac{(d_1+d_2+d_3+d_4)}{c}$
Geometry probability
$G(x_v) = \sum_{\forall x_l, x_s} H(x_l, x_s, t_v)$

Figure 2: Schematic view of the back-projection reconstruction for a single voxel of the hidden scene.

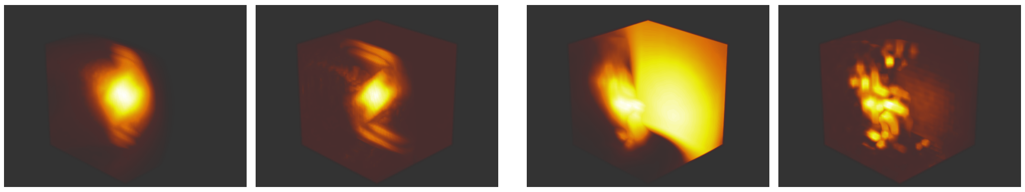

Figure 3: Examples of reconstructions of the Z and isolated bunny scenes.

of the MATLAB file. Figure 3 shows the four different hidden scenes whose NLOS measurements are provided in the datasets. All the datasets can be downloaded from the datset folder.

2.2 Basic tests and filtering

Test the implemented method over the `Z_d=0.5_l=[1x1]_s=[256x256].hdf5` dataset, and measure the time it takes reconstruct for different voxelization and capture grid resolutions. You can use the function `volshow` available on MATLAB along with the view configuration provided in the file `volshow_config.mat` to correctly display the reconstructed volume as `volshow(G,volshow_config)`.

You can also use the function `volumeViewer` of MATLAB, setting maximum intensity projection and the "hot" color map to have extra control over the visualization thresholds. Apply a Laplacian filter to the reconstructed volumes to improve the quality of the reconstructions. You can use the following code snippet to apply this filtering, assuming `G` is the 3D volume reconstructed with your back-projection method.

```
f_lap = fspecial3('lap');
G_lap = imfilter(G, -f_lap, 'symmetric');
```

2.3 Non-planar geometry

The `Z` dataset contains measurements for geometry contained within a single plane parallel to the relay wall. In contrast, the two `bunny_*` datasets contain measurements for an isolated non-planar geometry (the Stanford bunny) captured with different relay wall topologies: a single laser point 1×1 and a SPAD grid 256×256 in the file `bunny_d=0.5_l=[1x1]_s=[256x256].hdf5`, and an exhaustive scanning pattern with a laser grid (16x16) and a SPAD grid 16×16 in the file `bunny_d=0.5_l=[16x16]_s=[16x16].hdf5`. Additionally, the `bunnybox_d=0.5_l=[16x16] _s=[16x16].hdf5` dataset contains measurements of a Stanford bunny within a Cornell box, creating significant global illumination effects. Apply your reconstruction method to the different datasets and answer the questions asked later.

In your report, include visualizations of the reconstructed volumes from different viewpoints at different resolutions and with and without Laplacian filtering. Include the reconstruction time for each resolution you have used. Please answer the following questions and illustrate them with the obtained results when necessary:

1. How does the reconstruction time scale with respect to the hidden scene voxelization and the relay wall capture resolution?
2. How does reconstruction of non-planar geometry (e.g., the bunny scene) differ when using a single laser position and a SPAD grid versus combining laser and SPAD grids? What is the reason for the changes in the reconstructions?
3. Why do the outputs of back-projection reconstructions look blurry? What is the effect of applying a Laplacian filtering over this output and why does this happen?
4. What happens when reconstructing objects that are not isolated but surrounded by more geometry such as the bunny in a box? Try to explain why the reconstruction looks better or worse in this case, relating the resulting effects to the possible light paths with a specific time-of-flight.

Some aspects to consider when choosing reconstruction parameters:

- *Position and extent of the hidden geometry.* While in real applications, the extent of the target hidden geometry is unknown, the datasets in the supplemental material provide the position and dimensions of the region where the hidden geometry is contained. Namely, you can find this information in the dataset fields `volumePosition` and `volumeSize`. You can use this information to define the extent of your voxelization and avoid reconstructing large empty regions.
- *Resolution of the voxelization and capture grids.* Choose a tractable volume resolution to test your algorithm. A reconstruction grid of size $4 \times 4 \times 4$ will be fast to compute but will yield a very coarse reconstruction. A reconstruction grid of size $1024 \times 1024 \times 1024$ will provide a finer reconstruction but the algorithm will take forever to finish. A voxelization with resolution of $8 \times 8 \times 8$ or $16 \times 16 \times 16$ could be a good starting point to test that your algorithm works correctly. A resolution of $32 \times 32 \times 32$ may be a good trade-off between

reconstruction quality and execution time, once you know your method works well. For testing purposes, you can also try skipping part of the laser and SPAD points on the relay wall (e.g., one every two, one every four, on each dimension) at the risk of affecting the reconstruction quality.

3 Back-projection on confocal measurements

Confocal capture topologies synchronously co-locate lasers and SPADs when scanning the relay wall. Therefore, each illuminated point only generates one corresponding time-resolved measurement at the same location. Modify your back-projection method to handle confocal measurements, and apply it to the confocal version of the bunny scene `bunny_d=0.5_c=[256x256].hdf5`.

Please, include comparisons between the confocal reconstruction and the previous non-confocal reconstructions you made of the same scene. Is there any noticeable difference with reconstructions of non-confocal datasets?

Try to explain the observable differences in terms of the light transport paths that may take place in each capture topology.

4 Attenuation and foreshortening correction

You may have noticed the resulting reconstructions are attenuated towards regions far from the relay wall. This is caused partly by two factors:

1. Quadratic attenuation due to the different path distances leads to dimmer reconstructions at points of the hidden scene that are further away from the relay wall. This is specially noticeable in the scene containing two planes at different distances `planes_d=0.5_l=[16x16]_s=[16x16].hdf5`.

2. The foreshortening due to the surfaces cosine term introduces significant radial attenuation and, therefore, points away from the relay wall center present dimmer reconstruction values. This is specially noticeable in the `Z` scene.

Modify your back-projection algorithm to mitigate these attenuation effects given the known information about the reconstructed points and the normal at the relay wall for each laser and SPAD point. Hint: think about how quadratic and cosine attenuation occur in forward propagation of light paths (see Figure 2), and invert these operations when accumulating radiance back onto the voxels (see Equation (1)).

Illustrate the results in the `planes` and `Z` scenes and explain the motivation of the different factors you included in Equation (1) to mitigate the attenuation effects.

5 Phasor-based filtering

Reconstruct the hidden scene using a phasor-based 1D Morlet wavelet filter applied over the temporal domain of the H function before performing the backprojection operation, and compare the results with other filtered reconstructions you did before. This involves implementing this variation of the backprojection equation:

$$G(\mathbf{x}_v) = \left| \sum_{\forall \mathbf{x}_l, \mathbf{x}_s} H'(\mathbf{x}_l, \mathbf{x}_s, t_v) \right| \tag{2}$$

where

$$H'(\mathbf{x}_l, \mathbf{x}_s, t_v) = H'(\mathbf{x}_l, \mathbf{x}_s, t_v) *_t K_m(t), \tag{3}$$

where $*_t$ represents a convolution in the temporal domain, and

$$K_m(t) = e^{2j\pi\Omega_c t} e^{\frac{-t^2}{2\sigma^2}} \tag{4}$$

This filtering strategy is based on phasor-field NLOS imaging methods, and implements a complex-valued Morlet wavelet, i.e. a complex exponential with frequency Ω_c multiplied by a Gaussian envelope with standard deviation σ. To select values of Ω_c and σ follow this criteria

- Choose $\Omega_c = 1/\lambda_c$ so that the wavelength λ_c is at least twice the spacing between points (either vertical or horizontal) on the relay wall. Check the SPAD and laser grids to.
- Choose $\sigma \in [\frac{\lambda_c}{2\log 2}, 2\lambda_c]$.

Note this task not involve any major change of the backprojection algorithm. You only need to substitute the H evaluated in Equation (1) by its filtered counterpart H', and you only need to filter H once, before the actual algorithm starts. The indexing of the temporal domain in Equation (2) is the same as in Equation (1). Since the filtering introduces complex numbers in H, the reconstruction will also be complex-valued, so do not forget to apply an absolute value over the reconstruction before displaying it.

TIP 1: Since the value Δt in the datasets is defined as optical distance (in meter units), we recommend you formulate the 1D array t and the rest of the parameters Ω_c, σ in meter units too, assuming the domain t represents optical distance instead of the time of flight. This makes it easier to define values for $\Omega_c = 1/\lambda_c$ and σ based on the distance between points in the relay wall.

TIP 2: Performing a 1D convolution over the temporal domain of H using the `conv()` MATLAB function may require time-consuming iterations to perform individual temporal convolutions for every spatial coordinate of H. We recommend you apply convolutions as multiplications on the Fourier domain by first applying 1D Fourier transforms over the temporal dimension of H and K_m using `fft(variable, [], temporal_dimension)`, multiplying both transformed variables, and finally doing an inverse 1D Fourier transform `ifft(variable, [], temporal_dimension)` over the temporal domain to obtain the filtered H'.

6 Deliverable

Your deliverable should be an archive file (e.g. a ZIP file) with the following items:

1. All your MATLAB code, including commented code as well as a README file explaining how to use the code. Comment your design choices in the code, if any.
2. A PDF report with all the results (both intermediate and final) obtained, the explanations on how you got your results, and why you decided to use certain parameters or methods and not others.

7 Evaluation

The maximum score possible for this assignment is 10 points, but note that all parts sum to 12, so you can still get 10 points even if you do not get the maximum grade in all the parts:

Datasets (Section 2.1)	1
Basic tests and filtering (Section 2.2)	2
Non-planar geometry (Section 2.3)	2
Back-projection on confocal measurements (Section 3)	3
Attenuation and foreshortening correction (Section 4)	1
Phasor-based filtering (Section 5)	3
Total	12

References

[1] Miguel Galindo et al. “A dataset for benchmarking time-resolved non-line-of-sight imaging”. In: *ACM SIGGRAPH 2019 Posters* (2019), pp. 1–2. DOI: `10.1145/3306214.3338583`.

[2] Andreas Velten et al. “Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging”. In: *Nature Communications* 3.1 (2012), p. 745. DOI: `10.1038/ncomms1747`.