CS276 Assignment 3: NeRF for Sparse-view CT

Qihe Chen, 2020533088

December 17, 2022

1 Introduction

In this programming assignment, we implement basic FBP algorithm by Matlab[Tra21] and NeRF-based method by Pytorch[PGM+19].

2 Task A

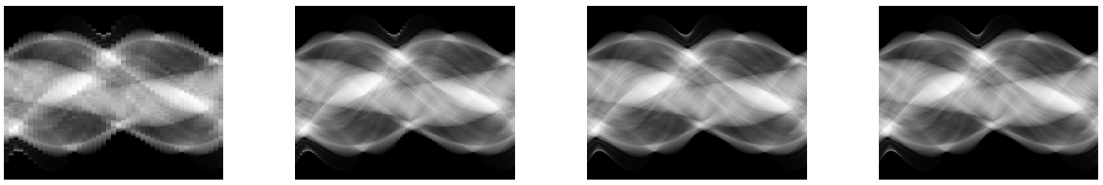
2.1 Dense-view sinogram generation: interpolated from sparse-view sinogram

We implement linear interpolation. To generate dense view, we upsample the sinogram to 360 number of angles, i.e. 1, 2, ..., 360. We implement the dense-view generation by linear interpolation.

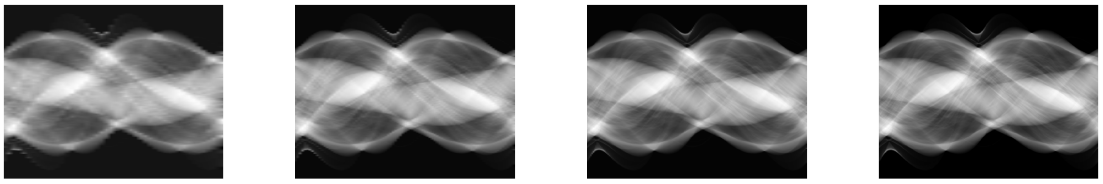
We define the original angles by $\mathbf{R}_s = [\mathbf{R}_s^i]_{i=1}^N$, it can be viewd as $N-1$ discritized regions [$\mathbf{R}_s^i$, $\mathbf{R}_s^{i+1}$]. For a new angle $\mathbf{R}_d^k$ to be interpolated, we find which region $\mathbf{R}_d^k$ locates at. And we interpolate the corresponding sinogram for $\mathbf{R}_d^k$ by

$$S_d^k = \frac{\mathbf{R}_s^{i+1} - \mathbf{R}_d^k}{\mathbf{R}_s^{i+1} - \mathbf{R}_s^i} S_s^i + \frac{\mathbf{R}_d^k - \mathbf{R}_s^i}{\mathbf{R}_s^{i+1} - \mathbf{R}_s^i} S_s^{i+1} \tag{1}$$

where S_d is the dense-view sinogram and S_s is the sparse-view sinogram.



(a) 45 sparse view (b) 90 sparse view (c) 180 sparse view (d) 360 sparse view



(e) 45 dense view (f) 90 dense view (g) 180 dense view (h) 360 dense view

Figure 1: The four images in the first row: different number of sparse views projected from ground truth source. The four images in the second row: upsampling 360 dense views from corresponding sparse view at the same column.

Given a sinogram S in a shape [**M**, **N**], where **M** is the number of sensors and **N** is the number of projected view or angles. In our implementation, we support several preprocessing or postprecessing

options, i.e. for preprocessing, we support passing the original sinogram by a simple triangle R—L filter (i.e. a kind of mid-pass filter), for postprocessing, we support passing the laplacian filter (i.e. a kind of second-order derivative filter). Between the preporcessing and the postprocessing, we implement the FBP algorithm and use the nearest neighbor interpolation for efficiency. We will discuss the details below.

2.2 Filtering backprojection algorithm: recover 2D CT source image from 1D arrays

We construct a 2D grid to store the reconstructed object in a shape [**M**, **M**] where **M** is the number of sensors as metioned above. Then we notice that each projection 1D array S^i only contributes a single pixel (x, y) at 2D grid once, i.e. we can traverse three for loops to calculte the once contribution of each S^i to each pixel (x,y).

The problem is how we can find which projection results on S^i should contribute to the pixel (x,y). By the definition of fan beam ray, it can be derived that

$$t = (x - \mathbf{M}/2)\cos(\mathbf{R}^i) - (y - \mathbf{M}/2)\sin(\mathbf{R}^i) + \mathbf{M}/2$$

where t is a floating point number. We thus use nearest neighor interpolation method to determine which projected result to be chosen, i.e. we round t into closest integer.

2.3 Filterng method

We need to perform FT on sinogram to transform sinogram from spatial domain to frequency domain. The frequency limit is chosen by finding the next power of 2 from M. There is a build-in useful matlab function, and the fft in matlab is also a build-in function

```
width = 2^nextpow2(M);
proj_fft = fft(sinogram, width);
```

For the preprocessing, we choose a simple triangle R—L filter, whose convolution kernel is defined by

```
kernel = 2*[0:(width/2-1), width/2:-1:1]'/width;
```

where the actual weight of this kernel shapes a triangle on the middle of the frequency domain (i.e. mid-pass filter). Since the convolution in spatial domain is exactly the multiplication in frequency domain, which is done by

```
proj_filtered = zeros(width,N);
for i = 1:N
    proj_filtered(:,i) = proj_fft(:,i).*filter;
end
```

For the postprocessing, we choose a laplacian filter, and the kernel is defined in spatial domain,

```
kernel = [0, 1, 0; 1, -4, 1; 0, 1, 0];
recon = imfilter(recon, kernel, 'replicate');
```

where **recon** is the reconstructed result image by FBP algorithm.

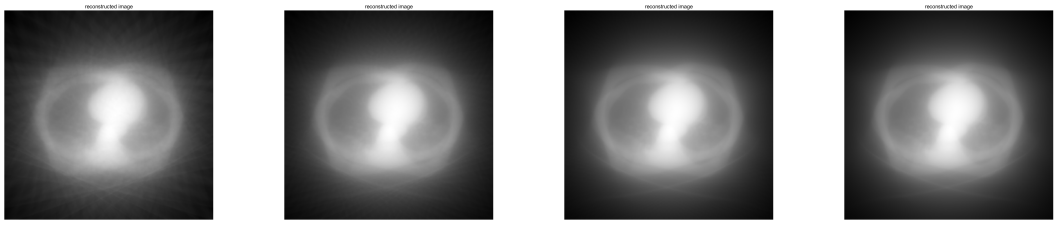


Figure 2: Sparse view 45∼360: no R-L filter, no laplacian filter

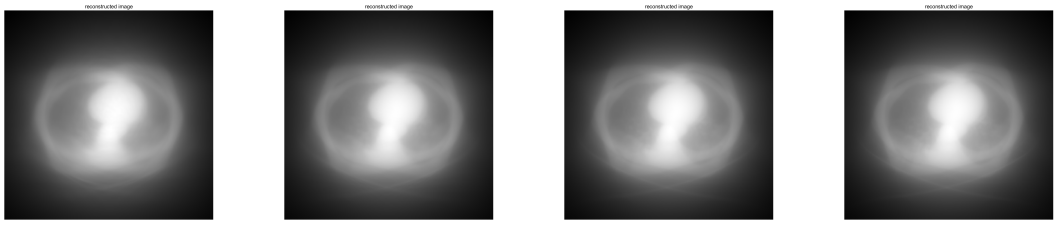


Figure 3: Dense view 45∼360: no R-L filter, no laplacian filter

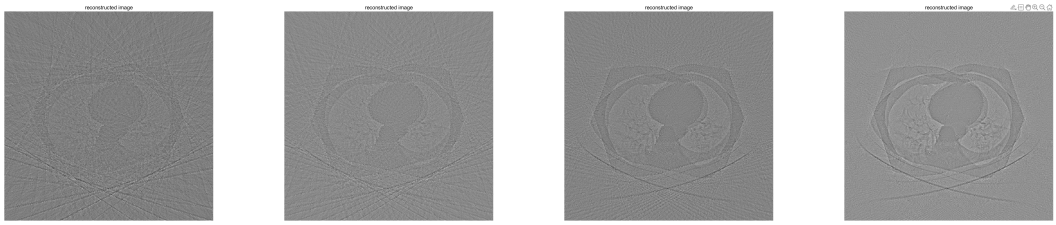


Figure 4: Sparse view 45∼360: no R-L filter, has laplacian filter

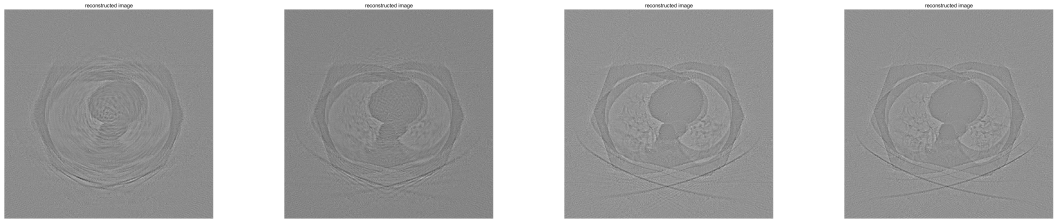


Figure 5: Dense view 45∼360: no R-L filter, has laplacian filter

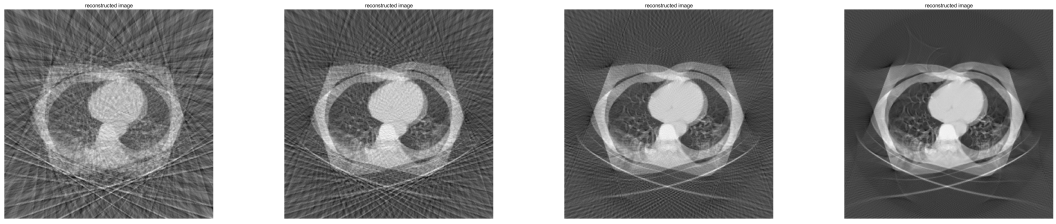


Figure 6: Sparse view 45∼360: has R-L filter, no laplacian filter

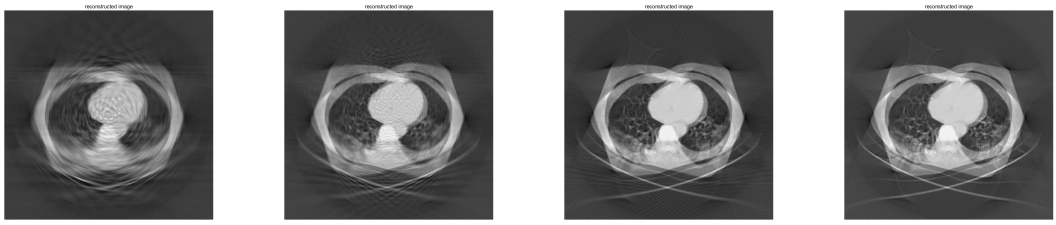


Figure 7: Dense view 45∼360: has R-L filter, no laplacian filter

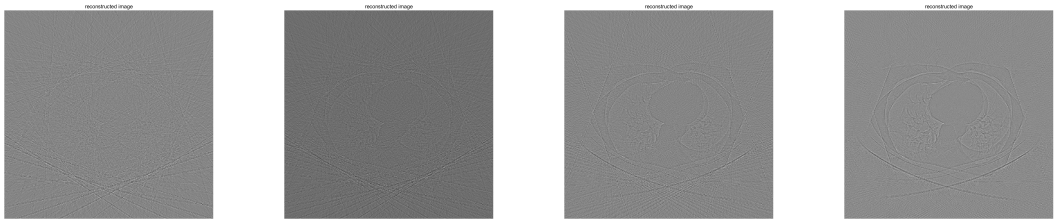


Figure 8: Sparse view 45∼360: has R-L filter, has laplacian filter

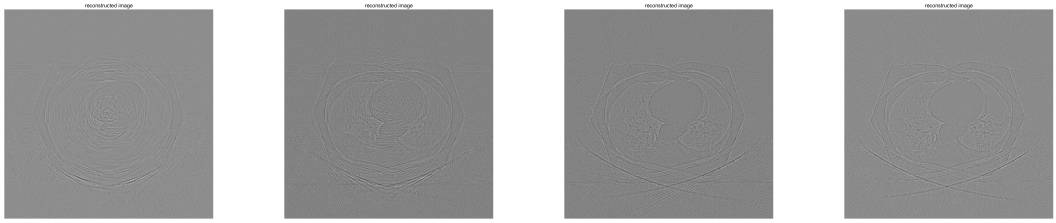


Figure 9: Dense view 45∼360: has R-L filter, has laplacian filter

3 Task B

We implement NeRF-based SVCT method. We modify the forward model (i.e.) volume rendering in original NeRF to orthographic projection (i.e. linear intergral or discritized sum). We will discuss about modified forward model detailly below. And we choose multiresolution hash encoding as our encoding method. Since we don't need to consider about view direction, we need not to find a encoding method for view direction. And we choose the activation of the **MLP** and density σ by Softplus.

3.1 Multiresolution hashing encoding

Muliresolution hash encoding has trainable encoding parameters θ, which can be jointly optimized with trainable weight parameter Φ of neural network. For example, given a fully connected neural network $F_\Theta(y;\Phi)$, multiresolution hash encoding outputs $y = \text{enc}(\mathbf{x};\theta)$ to improves the approximation quality and training speed across a wide range of applications without incurring a notable performance overhead. In this paper, we replace the positional encoding of original NeRF by multiresolution hash encoding to achieve completing training stage in a short time. For more details, please refer to prior work from MÜLLER [MESK22].

$$\mathbf{y} = \text{enc}(\mathbf{x};\theta) \tag{2}$$

We use 2D hashing encoding here, and for more specific details, please refer to network in our codes.

3.2 Forward model: orthographic projection

In computed tomography (CT) and our assignment setting, an orthographic projection is a one-dimensional array that is created by projecting the two-dimensional anatomy onto a flat line. To render a given projection result $S^i(x)$, where i is the i^{th} projection sinogram, x determines the position of sensor on this single sinogram array, we need to know all the incoming density σ by the ray $\mathbf{r}$ defined in NeRF.

$$\hat{S}^i(x) = \int_0^N \sigma(\mathbf{r}(t))dt \tag{3}$$

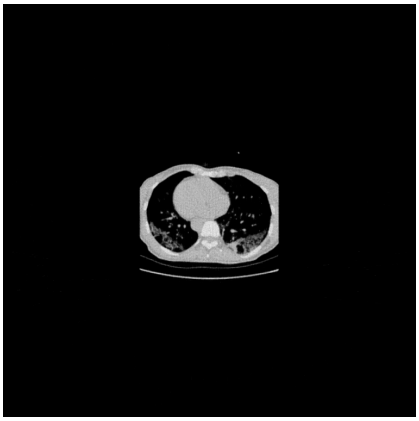
where $\hat{S}^i(x)$ is the approximate value given by NeRF, in practice, we sample several points $[r(t)]_{t=0}^{N-1}$ uniformly along the ray, and we perturb the points in training stage to give a better tolerance NeRF and we do not perturb them in testing stage. And then the linear intergral can been discritized by

$$\hat{S}^i(x) = \sum_{t=0}^{N-1} \sigma(\mathbf{r}(t))\delta_t \tag{4}$$

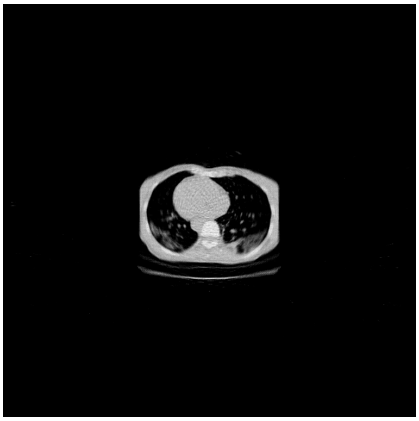
where $\delta_t = \mathbf{r}(t+1) - \mathbf{r}(t)$ determines the length of sample interval, here we assume the object between each sample interval is homogeneous. And we use MSELoss to update all the optimizable parameters.

(a) sparse view 45

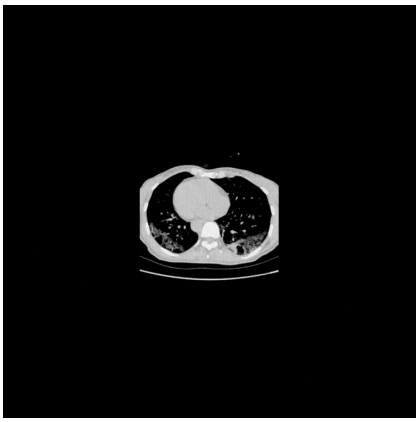
(b) dense view 45



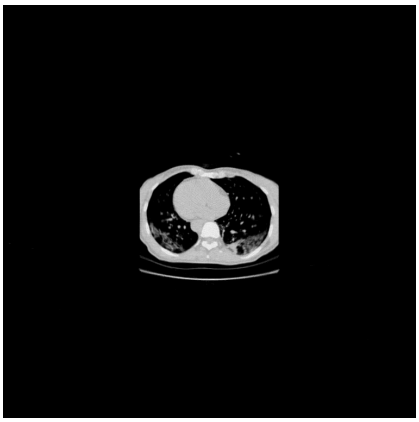
(c) sparse view 90



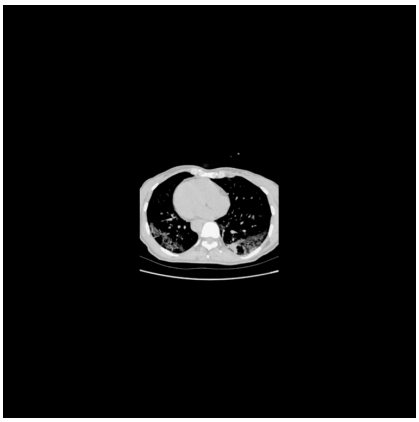
(d) dense view 90



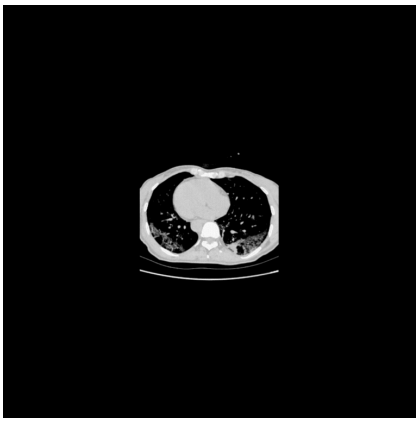
(e) sparse view 180



(f) dense view 180



(g) sparse view 360



(h) dense view 360

Figure 10: NeRF-based SVCT method render result

4 Task C

We first evaluate the reconstruction result from FBP and NeRF-based SVCT method respectively, and comapre their result to give our analysis.

4.1 FBP: sensitive to sparse-view input

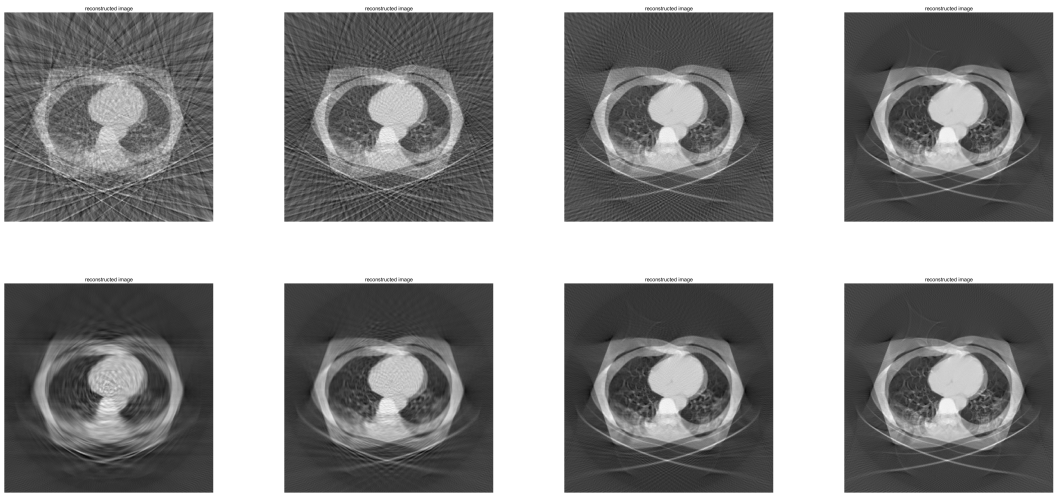
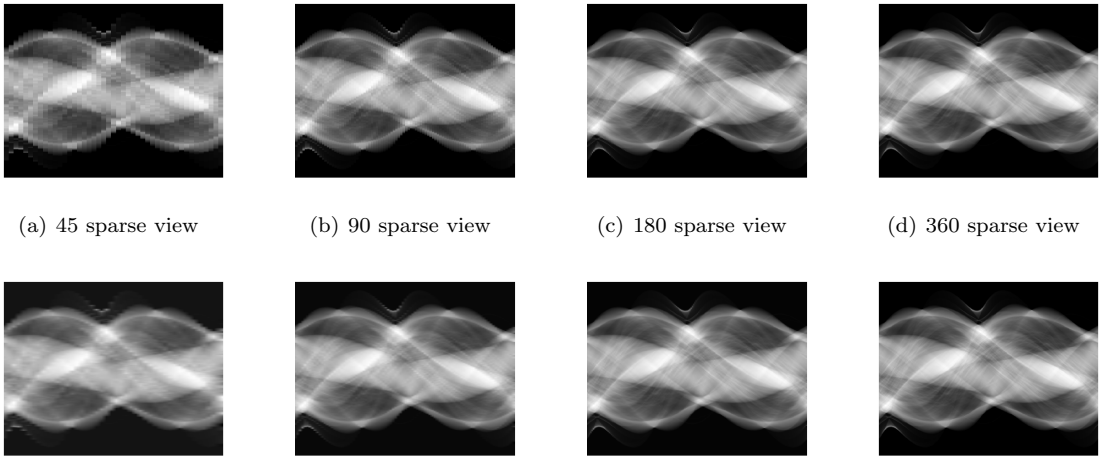


Figure 11: Sparse and dense view 45∼360: has R-L filter, no laplacian filter

In our implementation, without considering the number of views or upsamling dense-view, the FBP gives the best reconstruction results with R-L filter and without Laplacian filter.

1. If we upsample the original sparse view to generate dense view, the blurry upsampling sinogram will give a blurry FBP reconstruction result.



(a) 45 sparse view (b) 90 sparse view (c) 180 sparse view (d) 360 sparse view

(e) 45 dense view (f) 90 dense view (g) 180 dense view (h) 360 dense view

Figure 12: The four images in the first row: different number of sparse views projected from ground truth source. The four images in the second row: upsampling 360 dense views from corresponding sparse view at the same column.

2. If we increase the number of sparse view, the FBP algorithm will give us the increasing quality of reconstruction result.

In conclusion, 1. dense-view upsampling method can significantly improve the performance of FBP algorithm when the input sinogram is sparse-view. 2. FBP algorithm is sensitive to the number of projection arrays of input sinogram.

4.2 NeRF-based SVCT: not sensitive to sparse-input

As we shown previously, the reconstruction results do not have too many differences with the number of views increasing. Thus, we can say that NeRF with hash encoding is not sensitive to sparse-view. And we find that NeRF-based SVCT also has the problem that the reconstruction image becomes blurry when we use dense view.

Sparse view degrees/°	45	90	180	360
PSNR	47.50	53.81	59.73	60.66
SSIM	0.999831	0.999965	0.999993	0.999994

Table 1: NeRF-based SVCT: Sparse view reconstruction evaluation by PSNR and SSIM

Dense view degrees/°	45	90	180	360
PSNR	40.03	44.82	51.88	60.66
SSIM	0.999336	0.999798	0.999959	0.999994

Table 2: NeRF-based SVCT: Dense view reconstruction evaluation by PSNR and SSIM

5 Task 4

If we implement NeRF-like methods in frequency field, it will obey the learning habit of using multilayer perceptron (MLP). Generally, MLP tends to learn low-frequency part of input image and the high frequency part is always the hardest thing to be learned by MLP.

If we implement NeRF-like methods in frequency field, it will counter with sampling uniformly problem. NeRF needs sampling points along a single ray, which can cause biased problem in frequency domain. Since high frequency is harder learnable than low frequency, we apply uniformly weights to two imbalanced learnable things, which can cause biased problem and make the network harder to train. Also, the original sinogram distributes more on low frequency part due to the center Slice theorem, which can also cause biased problem.

If we implement NeRF-like methods in frequency field, it will counter with missing-wedge problem. More specificall, by center Slice theorem, the input sinogram has more low frequency part than high frequency part. And lots of high frequency part is unknown, which need interpolation method to fill up these missing-wedge. This will cause a lot of artefacts as the blurry dense-view reconstructed result we shown above.

References

[MESK22] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (TOG)*, 41:1 – 15, 2022.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems*, 2019.

[Tra21] Martin H. Trauth. Matlab® recipes for earth sciences. *MATLAB® Recipes for Earth Sciences*, 2021.