Assignment 1 Part 2: Neural Light Field Rendering

NAME: CHENQIHE
STUDENT NUMBER: 2020533088
EMAIL: CHENQH@SHANGHAITECH.EDU.CN

1 INTRODUCTION

In this assignment, we present a novel neural representaion for light field content and train the neural light field with lowtoys dataset. We use a fully-connected neural network and different types of encoders to get the pixel values from 4D light field coordinates.

There are more concrete problems that have been solved.

- Implement Neural light field renderer: LF_network. We use the combination of SIREN and Gegenbauer encoder as our own network architecture. And we try some other encoder like positional encoding and hashencoding, while Gegenbauer encoder performs the best effect to generate training views and novel views.
- We evaluation of our designs by showing the generated training views and novel views. We also show some results generated by other encoding methods.
- Implement the translational motion of the virtual viewing camera along x, y, and z directions.
- Implement the refocusing and change aperture size by reintroducing the effect of disparity.

2 RELATED WORK

In this section, we review related work to provide context for our own work.

Neural light field We design our neural light field by the combination with encoder and mlp network architecture. Crucially, we introduce a novel input transformation strategy of the multi-dimensional light field coordinate based on the orthogonal Gegenbauer polynomials, which performs the best quality for novel views synthesis. Furthermore, we test the same dataset named "lowtoys" with other encoders, like positional encodings.

Sine activation function Periodic nonlinearities have been investigated repeatedly over the past decades, but have so far failed to robustly outperform alternative activation functions. In this work, we implement Sine layer with sine activation function, and we use the initialization which is the same as in the SIREN.

Gegenbauer polynomials When we transform input into latent space and use neural network to learn a light field, it is commonly observed artifact in MRI reconstruction using Fourier-based approximations, which is commonly called "Gibbs phenomenon". Gegenbauer expansions provides a better convergence and usually resolves the Gibbs artifact using fewer basis functions than the Fourier approach.

Positional encoding Deep networks are biased towards learning lower frequency functions. Positional encoding can map the inputs to a higher dimensional space using high frequency functions before passing them to the network enables better fitting of data that contains high frequency variation by using some discretefourier basis.

Traditional light field It describes a simple and robust method for generating new views from abitrary(but actually limited) camera positions without depth information or feature matching, simply by combing and resampling the available images.

The key to this technique lies in interpreting the input images as 2D slices of a 4D function. The solution it proposes is to parameterize lines by their intersections with two planes in arbitrary position. In other words, it simplifies the plen-optic function from 7D into 4D.

3 METHODS

3.1 Neural light field renderer

Our network architecture is the combination of Gegenbauer encoder and MLP with Sine layer. We input 4D (u, v, s, t) coordinates into network, and get the output RGB values.

Given 4D light field coordinates $\mathbf{x} = (u, v, s, t) \in \mathbb{R}^{N\times 4}$, we first input them into our Gegenbauer encoder to transform inputs into higher dimensional latent space.

Gegenbauer polynomials

The nth-order Gegenbauer polynomials could be computed recursively as:

$$G_{n+1}^{(\alpha)}(x) = \frac{1}{n}[2x(n+\alpha-1)G_n^{(\alpha)}(x) - (n+2\alpha-2)G_{n-1}^{(\alpha)}(x)]$$

where $-1 \leq x \leq 1$, $G_0^{(\alpha)}(x) = 1$, and $G_1^{(\alpha)} = 2\alpha x$

To reconstruct pixels of a 4D light field, a coordinate-input vector $x = [u, v, s, t]$ is transformed with a set of functions S_i as

$$\epsilon(p) = [S_1(u), ..., S_{C_u}(u), S_1(v), ..., S_{C_v}(v), ..., \\ S_1(s), ..., S_{C_s}(s), S_1(t), ..., S_{C_t}(t)]$$

where C_u, C_v, C_s, C_t are the maximum orders of basis functions used to map the coordinates along each dimension.

In our case, we use the Gegenbauer basis functions to transform the input by setting $S_n(z) = G_n^{(\alpha)}(z)$.

In this work, we try $C_u = 16, C_v = 16, C_s = 240, C_t = 240, \alpha = 0.5$.

We adopt the sine activation presented by Sitzmann et al. as it enhances the ability of a MLP to approximate functions even without an input transformation.

student number: 2020533088
email: chenqh@shanghaitech.edu.cn

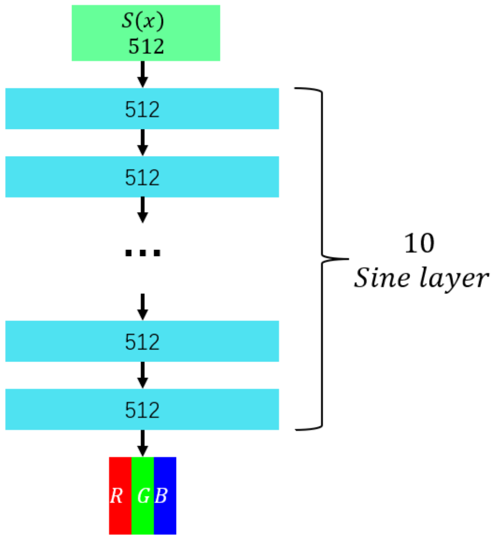
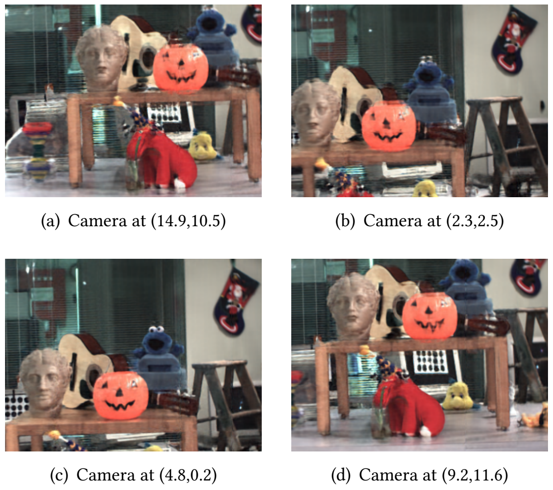
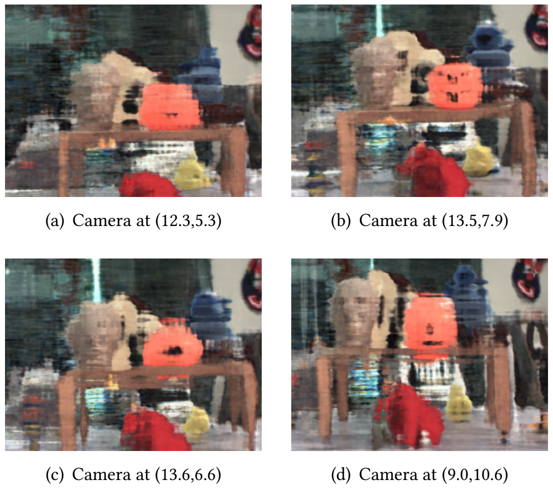


Fig. 1. A visualization of our fully-connected network architecture. Input vectors are shown in green, intermediate hidden layers are shown in blue, output vectors are shown in RGB, and the number inside each block signifies the vector's dimension. All layers are standard Sine layer in SIREN, black arrows indicate layers with sine activations, while the last balck arrow indicate layers with normalization. The Gegenbauer encoding of the input light field coordinates ($S(x)$) is passed through 10 fully-connected Sine layers, each with 512 channels. A final layer (with normalization) outputs the emitted RGB radiance at light field coordinate $x = (u, v, s, t)$, as viewed by camera (u, v) at pixel (s, t)



(a) Camera at (14.9,10.5) (b) Camera at (2.3,2.5) (c) Camera at (4.8,0.2) (d) Camera at (9.2,11.6)

Fig. 2. Novel views synthesis with Gegenbauer encoding



(a) Camera at (12.3,5.3) (b) Camera at (13.5,7.9) (c) Camera at (13.6,6.6) (d) Camera at (9.0,10.6)

Fig. 3. Novel views synthesis with positional encoding

3.2 Evaluation

We train our neural light field network with Gegenbauer encoder $\mathcal{G}$ for 1 day with 2038 epochs, and we show our final results for training views rendering and novel views rendering. We render the results for both training views and rendering views. Here, we only randomly generate 4 novel views for reference and the result can be not good due to the sparsity of the cameras.

We also try to replace the Gegenbauer encoding by positional encoding, and we set map each channel of $x = (u, v, s, t)$ into 20 channels, which results in 80 channels latent vector. And we use the same architecture same as the previous one instead of the encoder. Although positional encoding archieves high performance on training view rendering, the novel views rendering is really bad. We infer that the network architecture with 10 Sine layer is too deep for positional encoding. Thus, we can try some more shallow architecture to train our positional encoding.

We train positional encoding for 2 days with 3496 epochs, which gives good training views and bad novel views. Here, we generate 4 novel views.

In conclusion, Both of two encoding methods can render the training views very well, which implies that both methods can fit the training data very well. The Gegenbauer encoding can render novel views much better than positional encoding under the same deep network architecture.

3.3 Camera Motion

Without loss of generality, we implement the camera motion on the xy plane and along z axis.

For the camera motion on the xy plane, we can directly output the novel view at virtual camera (u, v).

For the camera motion along z axis, the situation becomes a little bit complex. However, there is a useful geometric similarity that can be used to generate the new image. We sample a ray per pixel.

Since the camera moves along the z axis, WLOG, we assume camera moves backward. Then the camera can see more scene containings than the camera on the xy plane. Thus, we implement the effect of increasing size of field of views by projection the region on the camera plane. And the ratio of region's width by height

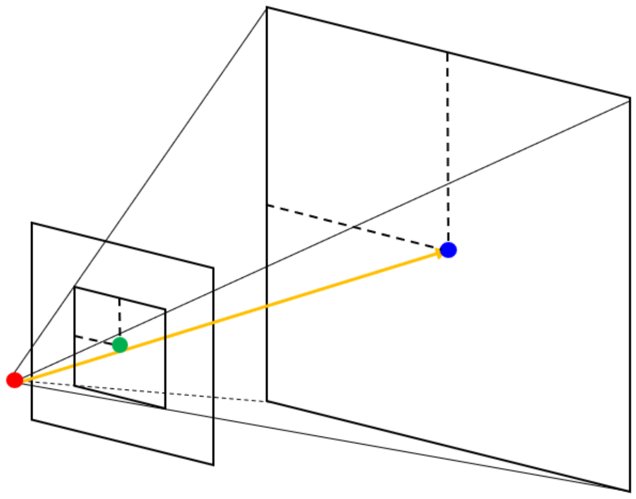
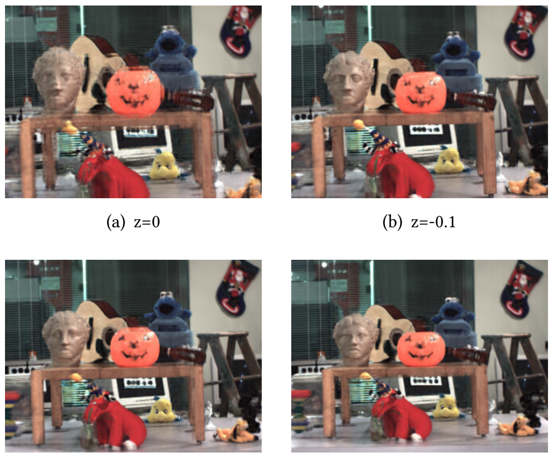


Fig. 4. A visualization about the geometric similarity. There are three parallelograms in the figure. The largest one on the right is the focal plane including the RGB pixels. The medium one on the left is the camera plane. The small one on the left is the view range for the camera moving along z axis. The red point is the moving camera, and we can get the value of pixels in the moving camera by sampling cameras at that projected region and getting the corresponding pixels at that sampled camera.

is the same as the image. And we sample cameras in the region just like how we sample pixels on the image. For each pixel, we sample a unique camera and the value of this pixel is valued as the corresponding pixel at that unique camera, which follows the geometric similarity.

We provide demo video. If you have any questions, please refer to my code for more details.



(a) z=0 (b) z=-0.1 (c) z=-0.2 (d) z=-0.3

Fig. 5. Camera moving along z axis with Gegenbauer encoding

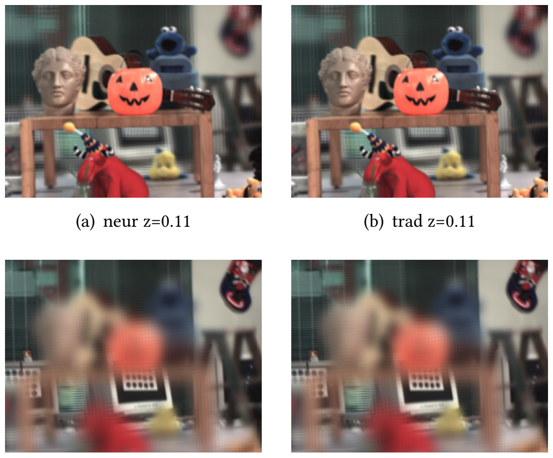
3.4 Refocusing

To implement the effect of refocusing, we establish a critical conception——disparity. Obviously, there are critical artifacts on the generated novel views. To implement correct effect of refocusing, we can only use the overfitting training views to blender the aperture views. Therefore, we can use the output of our neural light field, and for a given virutal camera, we implement the effect of aperture based on the disparity, which is the same as the traditional light field.

Therefore, we can regard our neural light field as the input light field data for the traditional light field. And we can use the pipeline of traditional light field to generate refocusing images.

And we compare our results with traditional light field.

The variable aperture size results are in the demo folder.



(a) neur z=0.11 (b) trad z=0.11 (c) neur z=0.20 (d) trad z=0.20

Fig. 6. Refocusing with Gegenbauer encoding