# Displaying Live 3-D Video from a Multi-View Camera on a Layered Display

**Yusuke Ota, Keita Maruyama, Ryutaroh Matsumoto, Keita Takahashi, Toshiaki Fujii**

Department of Information and Communication Engenieering, Graduate School of Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya, Aichi 464-8603, Japan

Keywords: layered display, multi-view images, convolutional neural network, multi-view camera

**ABSTRACT**

*We present a pipeline that displays 3D videos captured by a multi-view camera (ProFUSION25) on a layered display in real time. The layered display is a kind of light field displays. To develop this pipeline, we used a CNN that calculates a layer pattern to reduce processing time.*

## 1. INTRODUCTION

A layered display is one of 3D displays that can be viewed in 3D with naked eyes [1]. Our prototype layered display is shown in Figure 1, in which several liquid crystal panels are stacked in front of the backlight and their transmittance can be controlled pixelwise [2]. According to the viewing direction, the overlapping of the pixels changes, a different image can be viewed, and a 3D image can be displayed. To display a 3D image, it is necessary to optimize the layer pattern, which is composed of the transmittance of each pixel so as to reproduce the input multi-view images as accurately as possible.

In our previous work [2], we calculated the layer pattern using a non-negative tensor factorization (NTF) as was proposed in [1]. Moreover, when using a capturing device with wide viewpoint intervals, such as a multi-view camera, we revealed that it is necessary to generate higher density multi-view images by viewpoint interpolation. However, it took a long time to conduct the viewpoint interpolation and calculate the layer pattern with the NTF, and real-time display of the captured image was difficult.

Therefore, we proposed a convolutional neural network (CNN) that performs viewpoint interpolation and a layer pattern calculation collectively [3], and conducted an experiment on the Stanford new light field dataset [4]. We confirmed that the calculation method using the CNN is much faster than the method using the NTF, and almost the same image quality can be achieved. In our subsequent study [5], using that network [3], we confirmed that sufficient image quality could be obtained for a still image captured by a multi-view camera ProFUSION25 [6]. In the present study, we are developing a pipeline that can display 3D videos in real time on the layered display from a multi-view camera ProFUSION25. As far as we know, this is the first pipeline capable of real-time display of 3D videos on a layered display.

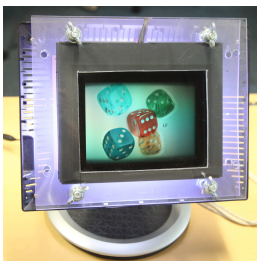


Fig. 1: Layered display

Fig. 2: ProFUSION25

Fig. 3: Multi-view images

Table. 1: Specification of ProFUSION25

| | |
|---|---|
| Resolution | $640 \times 480$ pixel |
| Frame rate | 25 fps |
| Number of viewpoints | 25 views |
| Output format | 8 bit |
| Size | $90 \times 90 \times 60$ mm |

## 2. CALCULATION OF LAYER PATTERN FROM CAPTURED IMAGE

In Section 2, we describe the proposed pipeline.

### 2.1 Shooting with ProFUSION25

In this paper, we use a ProFUSION25 to acquire an object to be displayed on the layered display. Figures 2, 3 and Table 1 show the appearance, the multi-view images captured by the ProFUSION25 and the specification, respectively. The ProFUSION25 is a 25-eye camera array system capable of capturing multi-view video into a PC in real time. This camera can capture the $5 \times 5$ multi-view images in a single shot and the frame rate is 25 fps.

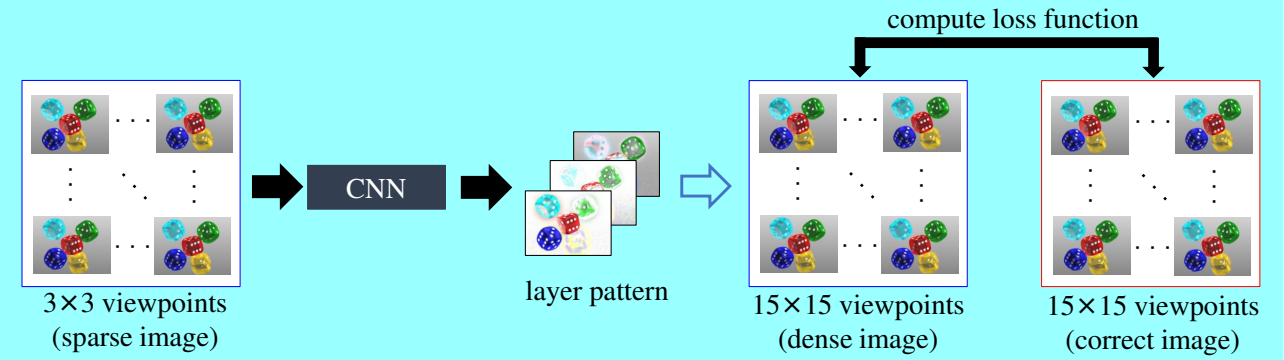



Fig. 4: Learning flow

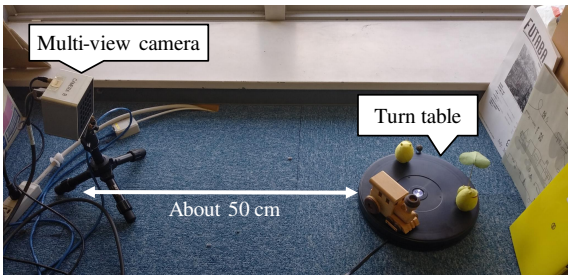



Fig. 5: Shooting environment

### 2.2 Generation of multi-view images as input to a CNN

We perform two steps to generate the multi-view images that will be input to the CNN. The ProFUSION25 has a structure in which $5 \times 5$ cameras are arranged in parallel, but the actually-taken multi-view images are not exactly arranged. Therefore, we performed rectification to create a virtually parallel state by image processing. In this paper, we used the method of Fukushima et al. [7]. In addition, the multi-view images captured by multiple cameras installed in parallel have a convergence plane at infinity. The layered display displays the convergence plane at the center layer. So the convergence plane of the input multi-view images is adjusted to a certain distance in the target scene [2].

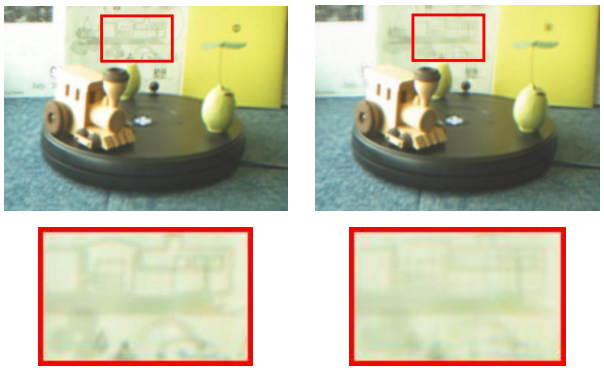
### 2.3. Calculation of layer pattern using a CNN

To calculate the layer pattern, we applied our method [3] of optimizing the layer pattern to reproduce a dense multi-view images from a sparse multi-view images using a CNN. The learning flow is shown in Figure 4. The CNN consists of 4 layers, which are convolutional layers connected in series. The relu fuction is used as the activation for the 1st to 3rd layers, and the sigmoid function is used as the activation for the last layer.

For the input to the CNN, we used the central $3 \times 3$ views of the $5 \times 5$ views captured by the ProFUSION25. The loss function was calculated from the multi-view images reproduced by the layer pattern and the ground-truth multi-view images. As the ground-truth multi-view images, we used the virtual $15 \times 15$ multi-view images obtained by applying viewpoint interpolation to $3 \times 3$ views. This is because a high-quality layer pattern can be generated from the sparse multi-view images by using the viewpoint-interpolated dense multi-view images as the ground-truth [2]. The layer pattern was optimized to reproduce the ground-truth $15 \times 15$ views.

Table. 2: Comparison of calculational time and PSNR

| | Layer calculation[sec] | PSNR [dB] |
|---|---|---|
| NTF [2] | 11.7 | 33.74 |
| CNN [3] | 0.03 | 30.50 |



(a) NTF (b) CNN

Fig. 6: Experimental result

### 2.4. Parallel processing

To speed up the process, we parallelized tasks in 5 steps using multiple CPU cores. The 1st step is to transfer data from the ProFUSION25 to the PC using DMA (direct memory access). In the 2nd and the 3rd steps, rectification and adjustment of convergence plane are performed for 5 views and 4 views using CPU, respectively. In the 4th step, the layer pattern is calculated from the processed 9 views using GPU and CPU. In the 5th step, calculated layer pattern is displayed on the layered display. To display the calculated layer pattern, we used the OpenCV

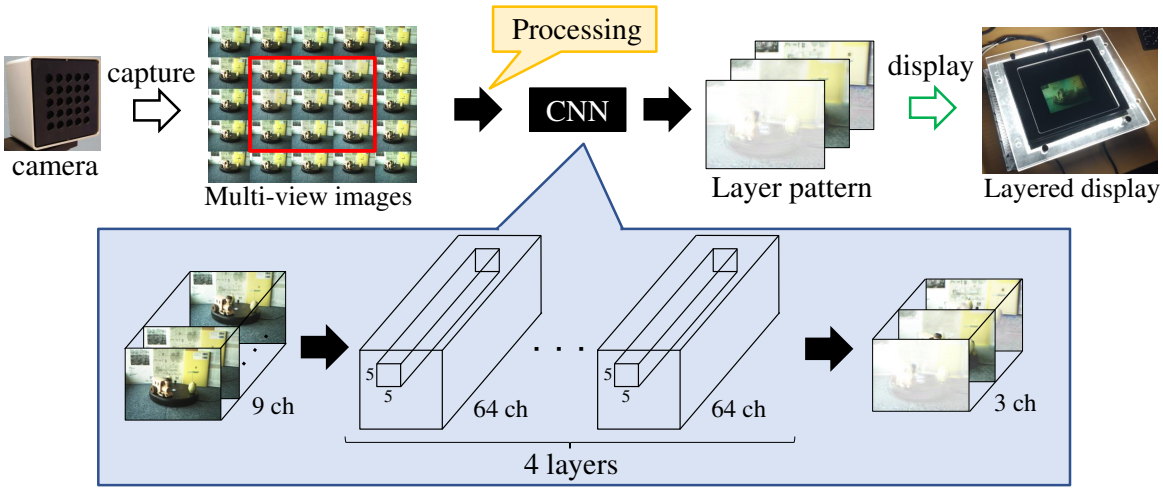



Fig. 7: Process flow from capturing to displaying

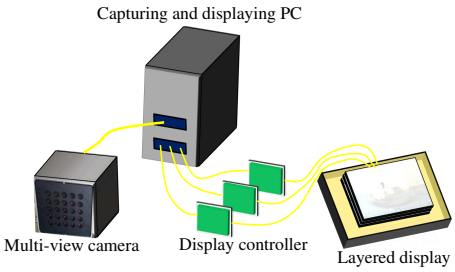



Fig. 8: Overall system

Fig. 9: Displayed images

function imshow. Since each step is performed in parallel, the other processes can be performed while transferring data from the ProFUSION25 to the PC, and the slowest step among the 5 steps decides the frame rate.

## 3. EXPERIMENTS

We used a PC running an open source software Ubuntu 18.04 LTS, equipped with an Intel Core i5-4590 3.30 GHz central processing unit (CPU), 16 GB main memory, a GeForce GTX 1080 Ti graphics processing unit (GPU), TensorFlow 1.12, and Python 3.6.

First, the layer pattern was calculated from still multi-view images. We used the 10 kinds of the multi-view images captured by the ProFUSION25 as training data. The patch size was $32 \times 32$, the 6000 patches were randomly generated for each dataset, and we trained the CNN for 50 epochs. The mean squared error and Adam were used for the loss function and optimization method, respectively.

The experimental environment is shown in Figure 5, and Table 2 shows the time required to calculate layer pattern and the image qualities using the CNN and NTF. We iterated the multiplicative update in the NTF for 50 times. The quality was evaluated by PSNR between the $15 \times 15$ multi-view images reproduced by the obtained layer pattern and the virtual $15 \times 15$ multi-view images obtained by viewpoint interpolation [8]. Figure 6 shows the central view image of the $15 \times 15$ views reproduced by the obtained layer pattern (simulated image). From the results in the Table 2 and Figure 6, the method using the CNN ran faster for calculating the layer pattern than the NTF, and could achieve almost the same quality as the NTF. In addition, we succeeded in improving PSNR by 8 dB compared to [5].

Next, we performed an experiment to calculate the layer pattern from multi-view videos. The procedure from capturing multi-view images to displaying on the layered display and the diagram are illustrated in Figures 7 and 8. The $5 \times 5$ multi-view images captured by the ProFUSION25 are transferred to the PC. Rectification and adjustment of convergence plane are performed on the central $3 \times 3$ views, and an appropriate layer pattern is calculated from the processed multi-view images using the CNN and displayed on the layered display. The ProFUSION25 is connected to a PCI card. The graphic board equipped with our PC has 4 synchronized HDMI outputs, from which the video signals are fed to the layered display via dedicated controller circuits.

The displayed images on the display using the above system are shown in Figure 9. The frame rate in each step is shown in Table 3. We confirmed that the images dis-

Table. 3: Frame rate in each step

| | frame rate [fps] |
|---|---|
| 1st step | 25.17 |
| 2nd step | 22.49 |
| 3rd step | 24.73 |
| 4th step | 18.32 |
| 5th step | 18.29 |

played on the layered display are observed in 3D and our network is applicable to capture and display. The frame rate of the video displayed on the layered display was about 18 fps. In the 4th step, it took a longer time than Table 2 because of the concatenation of the multi-view images processed in the 2nd step and 3rd step.

### 4. CONCLUSION

We developed a 3D display system from capturing multi-view images to displaying on the layered display. In the conventional method, it took a long time to calculate the layer pattern, but by using the CNN that we proposed, calculational time was reduced to about $1/400$ and the image quality was almost the same compared to the NTF. Real time display that was impossible with the conventional method was achieved using this CNN and parallelizing the 5 steps from capturing to displaying. For future work, we will improve the layer pattern quality and calculational time.

## References


[1] G. Wetzstein et al., "Tensor Displays: Compressive Light Field Synthesis using Multilayer Displays with Directional Backlighting," ACM TOG, vol. 31, no. 4, article ID 80, 2012.

[2] Y. Kobayashi et al., "A 3-D Display Pipeline: Capture, Factorize, and Display the Light Field of a Real 3-D Scene," ITE Trans. on MTA, vol. 5, no. 3, pp. 88–95, 2017.

[3] Y. Ota et al., "3-D Display from a Sparse Multi-View Camera to a Layered Display using CNN," The 34 th PCSJ and 24 th IMPS, pp.166–167, 2018 (In Japanese).

[4] V. Vaish et al., "The (New) Stanford Light Field Archive," http://lightfield.stanford.edu/lfs.html, 2008.

[5] Y. Ota et al., "Displaying Real 3D Object Taken from Multi-View Camera on Layered Display Using Deep Learning," 3D Image Conference, 2019 (In Japanese).

[6] D. Sekiguchi et al., "Development of ProFUSION25," Information Processing Society of Japan (IPSJ), vol. 2008, no. 27, pp.239–242, 2008.

[7] N. Fukushima et al., "Rectification Method for Two-dimensional Camera Array by Using Parallelizing Locus of Feature Points," The journal of the Institute of Image Information and Television Engineers, vol. 62, no. 4, pp.564–571, 2008.

[8] N. K. Kalantari et al., "Learning-based View Synthesis for Light Field Cameras," ACM TOG, vol. 35, no. 6, pp.193:1–193:10, 2016.