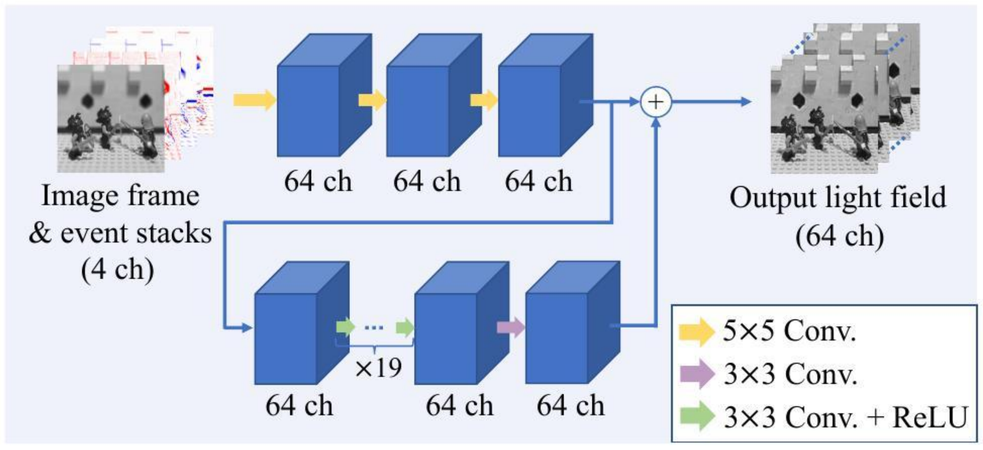


Figure 2. Network architecture of RecNet

and $a^{(3)}$, respectively. In line 9, we compute an event stack from the consecutive coded-aperture images as

$$E_{x,y}^{(n,n+1)} = Q\left(\frac{\log(I_{x,y}^{(n+1)} + \epsilon) - \log(I_{x,y}^{(n)} + \epsilon)}{\tau + n_{x,y}}\right) \quad (12)$$

where $Q(x) := \mathrm{sign}(x)\mathrm{floor}(|x|)$ is a quantization operator, and $\epsilon = 0.01$ w.r.t. the intensity range of $[0, 1]$ for $I^{(n)}$ and $I^{(n+1)}$. To account for the randomness of the sensor, a zero-means Gaussian noise $n$ with $\sigma = 0.021$ is added to the contrast threshold $\tau$. Equation (12) follows a widely used event simulator [36], but we implement it as being differentiable. Following previous studies [14, 28, 42], we also add a zero-means Gaussian noise ($\sigma = 0.005$ w.r.t. the normalized intensity range $[0, 1]$) to the image frame $\bar{I}$ to account for the measurement noise.

A challenging issue with Eq. (12) is how to determine the contrast threshold $\tau$. When the imaging process is computationally simulated, a smaller $\tau$ is better. This is because a smaller $\tau$ leads to a more fine-grained observation (a larger number of events), which in turn results in more accurate light-field reconstruction. However, it is difficult to determine a specific $\tau$ that is compatible with a real event camera (e.g., a DAVIS 346 camera) the $\tau$ of which cannot be controlled in an explicit (direct) manner. With our experimental configuration, we empirically estimate $\tau \simeq 0.15$, but it depends on the configuration. To address this issue, we treat $\tau$ as being variable, which makes the model of Eq. (12) more flexible. Specifically, we randomly draw $\tau$ from $[0.075, 0.3]$ for each batch during training, which enables the algorithm to be independent of a specific $\tau$.

**RecNet**. Since network architectures are not the main focus of this paper, we used a plain network architecture consisting of a sequence of 2-D convolutional layers. The data obtained from the camera ($\bar{I}$, $E^{(1,2)}$, $E^{(2,3)}$, and $E^{(3,4)}$ stacked along the channel dimension) are fed to RecNet; the output from RecNet is a tensor with 64 channels corresponding to 64 views of the reconstructed light field (see Fig. 2 for more details). Similar architectures were used in previous works [14, 42], achieving a good balance between the reconstruction quality and computational cost. Our plain architecture can be used for other imaging methods with minimal modifications (only by changing the number of channels for the input layer), which makes the comparison easier. Exploration for better network architectures is left as future work.

**Training and loss function**. AcqNet and RecNet are implemented using PyTorch 2.0 and jointly trained on the BasicLFSR dataset [23]. We extract 29,327 training samples, each with $64 \times 64$ pixels and $8 \times 8$ views, from 144 light fields designated for training. The disparities among the neighboring viewpoints are mostly limited within $[-3, 3]$ pixels. This is suitable for our method because the $8\times8$ viewpoints are arranged on the small aperture plane of the camera, which results in limited disparities among the viewpoints. We use the built-in Adam optimizer with default parameters and train the entire network over 600 epochs with a batch size of 16. The scale parameter $s$ is initialized as 1 and multiplied by 1.02 for each epoch.

As mentioned earlier, we use the reconstruction loss between the original and reconstructed light fields. However, this is insufficient in some cases. The learned patterns sometimes have significantly different brightness (the brightness of $a^{(n)}$ is given as $\sum_{u,v} a_{u,v}^{(n)}$), which causes too many events spreading over all the pixels. These coding patterns are useless with a real event camera because such a large number of events cannot be recorded correctly due to the limited throughput of the camera. To avoid this problem, we optionally add the second term to the loss function to suppress the number of events in each batch ($N_{\mathrm{event}}$) below a pre-defined threshold ($\theta$).

$$\mathrm{Loss} = \mathrm{MSE}(L, \hat{L}) + \lambda \cdot \max(N_{\mathrm{event}} - \theta, 0) \quad (13)$$

where $\lambda$ is a non-negative weight. The threshold $\theta$ can be computed from the camera's throughput (events per sec.).

### 3.4. Hardware

Our hardware setup is shown in Fig. 3. The optical system consists of a Nikon Rayfact lens (25 mm F1.4 SF2514MC), set of relay optics, beam splitter, and LCoS display (Forth Dimension Displays, SXGA-3DM, $1280\times1024$ pixels). We use an iniVation DAVIS 346 monochrome camera that can acquire both events and image frames with $346\times260$ pixels.

We use the central portion of the LCoS display ($1024 \times 1024$ pixels) as the effective aperture area, and divide it into $8 \times 8$ regions (each with $128 \times 128$ pixels) to display the coding patterns (each with $8 \times 8$ elements). Four coding patterns, $a^{(1)}$, $a^{(2)}$, $a^{(3)}$, and $a^{(4)}$, are repeatedly displayed, each with a 5.0 msec duration. According to the hardware's document, the total time for each coding pattern ($T_{\mathrm{c}}$) takes 5.434 msec including the overhead time. The exposure time for an image frame $\bar{I}$ is set to $4T_{\mathrm{c}}$ ($\simeq$ 22 msec[3]) to cover a

[3]This is shorter than those in some previous studies; the exposure time of a coded-aperture camera was set to 40 msec [14, 38] and 68 msec [28].