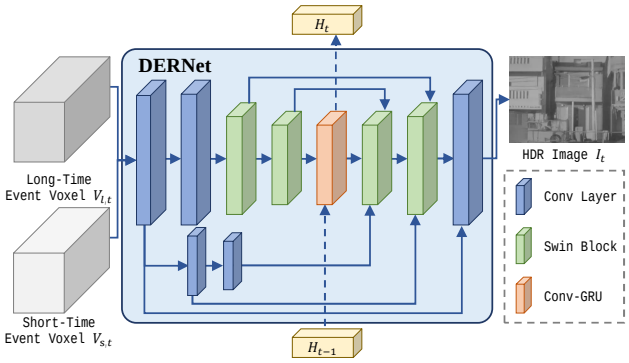

Figure 1. Architecture of DERNet

1. Overview

To achieve high-speed HDR video reconstruction from events, our team introduces the Dual Event-stream Reconstruction Network (DERNet). As depicted in Figure 1, DERNet uses long-time and short-time event voxels to reconstruct the low-frequency brightness and high-frequency texture of HDR video. Furthermore, DERNet integrates Swin Transformer and Conv-GRU blocks to capture spatial and temporal contexts, thereby enhancing reconstruction accuracy.

2. Network Architecture

DERNet adopts an encoder-decoder network with a recursive design to process dual-stream event voxels to estimate high-speed HDR videos. Specifically, when reconstructing the t-th frame of the HDR video, considering that the long-time event stream around frame t can help reconstruct the low-frequency brightness, DERNet voxelizes the event data from frame $t - T_l$ to frame $t + T_l$ into a b-bins event voxel $V_{l,t}$. Simultaneously, considering that the short-time event stream around frame t can help reconstruct high-frequency texture, DERNet voxelizes the event data from frame $t - T_s$ to frame $t + T_s$ into a b-bins event voxel $V_{s,t}$. Subsequently, the event voxels $V_{l,t}$ and $V_{s,t}$ are concatenated and input into the network. To fuse the features of the two event voxels, DERNet utilizes convolutional layers to generate fused features from the event voxels. The network then adopts a two-branch encoder. This structure includes a complex branch that extracts high-level semantic information from the fused features, leveraging Swin Transformer blocks to capture spatial context and Conv-GRU blocks to capture temporal context by integrating historical states. It also includes a simple branch that utilizes convolutional layers to capture detailed information from the fused features. Next, the decoder of DERNet adopts multiple Swin Transformer blocks to fuse and upsample the features extracted by the two-branch encoder, finally using convolutional networks to predict the t-th frame HDR image I_t.

3. Loss Function

To train DERNet, a reconstruction loss $\mathcal{L}_r$ is designed for the estimated HDR image $\mathcal{L}_t$ as

$$\begin{aligned} L_r = \lambda_1 \mathrm{L}_1(I_t, I_t^{gt}) + \lambda_2 \mathrm{L}_1(M(I_t), M(I_t^{gt})) \\ + \lambda_3 \mathrm{L}_2(I_t, I_t^{gt}) + \lambda_4 \mathrm{L}_2(\mathrm{M}(I_t), M(I_t^{gt})) \end{aligned} \tag{1}$$

where λ_1, λ_2, λ_3, and λ_4 coefficients balancing the loss terms, $\mathrm{L}_1(\cdot,\cdot)$ is the absolute loss function, $\mathrm{L}_2(\cdot,\cdot)$ is the mean squared error loss function, I_t^{gt} is the ground truth t-th frame HDR image, and $\mathrm{M}(\cdot)$ is the HDR to SDR function defined as $\mathrm{M}(x) = \frac{\log(1+5000x)}{\log(5001)}$.

4. Implementation Details

DERNet is implemented using PyTorch. During training, a batch size of 2 is utilized, with a video sequence length of 10 and a data size of 224×224. An AdamW optimizer is adopted with a learning rate of 4×10^{-5} and weight decay of 10^{-6} to optimize the network weights for 60 epochs. A cosine annealing scheduler is adopted to decay the learning rate. To prevent overfitting, random flipping, rotation, and cropping are applied to the event voxels for data augmentation. The coefficients are defined as $b = 6$, $T_l = 16$, $T_s = 5$, $\lambda_1 = 1$, $\lambda_2 = 0.1$,$\lambda_3 = 500$, and $\lambda_4 = 10$.