

## SUPPLEMENTARY MATERIAL

We present implementation details and additional experiments in our supplementary material. Please watch the accompanying videos with 0.5 speed to see the artifacts generated by the different methods.

### A TECHNICAL DETAILS

#### A.1 VISUAL REFERENCE FOR STEREOEFFECTS

We provide a visual reference for the optical flow analysis as below:

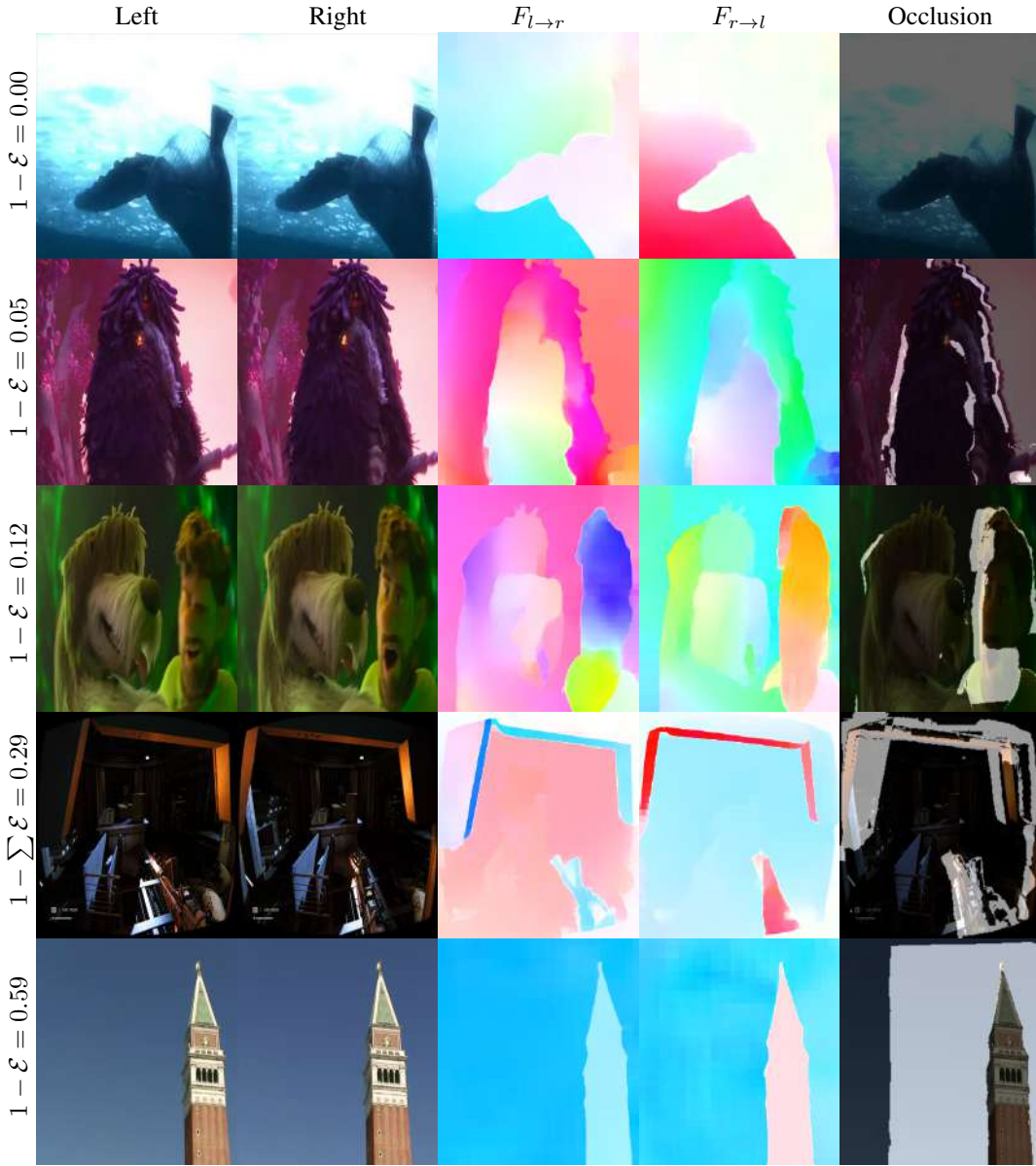


Figure 6: Visual demonstration of the different levels of stereo effects with the dataset.

## A.2 CONTEXT ENCODER

Our context encoder uses a stack of convolutional layers to obtain semantic features from the input image as below. Starting from the 5<sup>th</sup> convolution layer, the extracted features are the concatenation of the features from the current and previous layers.

```
Conv2d(3, 64, kernel_size=3, stride=1, padding=1),
LeakyReLU(0.2, inplace=True),
Conv2d(64, 64, kernel_size=3, stride=2, padding=1),
LeakyReLU(0.2, inplace=True),
Conv2d(64, 128, kernel_size=3, stride=1, padding=1),
LeakyReLU(0.2, inplace=True),
Conv2d(128, 256, kernel_size=3, stride=1, padding=1),
LeakyReLU(0.2, inplace=True),
Conv2d(256, 384, kernel_size=3, stride=1, padding=1, groups=1),
LeakyReLU(0.2, inplace=True),
Conv2d(384, 512, kernel_size=3, stride=1, padding=1, groups=2),
LeakyReLU(0.2, inplace=True),
Conv2d(512, 384, kernel_size=3, stride=1, padding=1, groups=4),
LeakyReLU(0.2, inplace=True),
Conv2d(384, 256, kernel_size=3, stride=1, padding=1, groups=8),
LeakyReLU(0.2, inplace=True),
```

## B ADDITIONAL EXPERIMENTS

### B.1 ALTERNATIVE MASK SELECTION ALGORITHM

To avoid having multiple pixels being mapped to the same pixel location  $i, j$ , we use an algorithm to produce  $[0, 1]$  masks so that different layers cannot interfere with each other as shown in Algorithm 1. In addition, we further tested another design where the mask value selection algorithm Algorithm 2 generates mask values  $\in \{-1, 0, 1\}$ , to allow more interactions between layers. However, though Algorithm 2 can better resolve complicated scenarios, we found the intermediate implicit disparity layers often fail to resolve disparities correctly, as shown in Figure 7. In general, we found that the Algorithm 2 tries to weaken the disparity cues, resulting in smoother output with weaker or wrong disparity maps.

---

#### ALGORITHM 2: Synthesis from layered disparities.

---

```
# number_layered_disparity: the number of disparity layers.
# warped_output: 'BDTCHW'. A stack of images warped by layered
# disparities. D is the number of disparity layers.
# warped_mask: 'BDTCHW'. A stack of masks warped by layered disparities.
# D is the number of disparity layers.
layered_mask = zeros_like(output_mask)
total_mask = zeros_like(output_mask)
for i in range(number_layered_disparity):
    if i == 0:
        layered_mask[:, i] = warped_mask[:, i]
        total_mask[:, i] = warped_mask[:, i]
    else:
        total_mask[:, i] = logical_or(warped_mask[:, i], layered_mask[:, i - 1])
        layered_mask[:, :, i] = total_mask[:, :, i] - output_mask[:, :, i - 1]
output = layered_mask * warped_output
```

---

### B.2 CONTEXT FUSION MODULE

As shown in Table 3, the inclusion of the context fusion module significantly enhances the overall statistical performance. Moreover, as demonstrated in the accompanying videos, this module greatly improves the temporal consistency of the generated videos. However, we observed potential artifacts

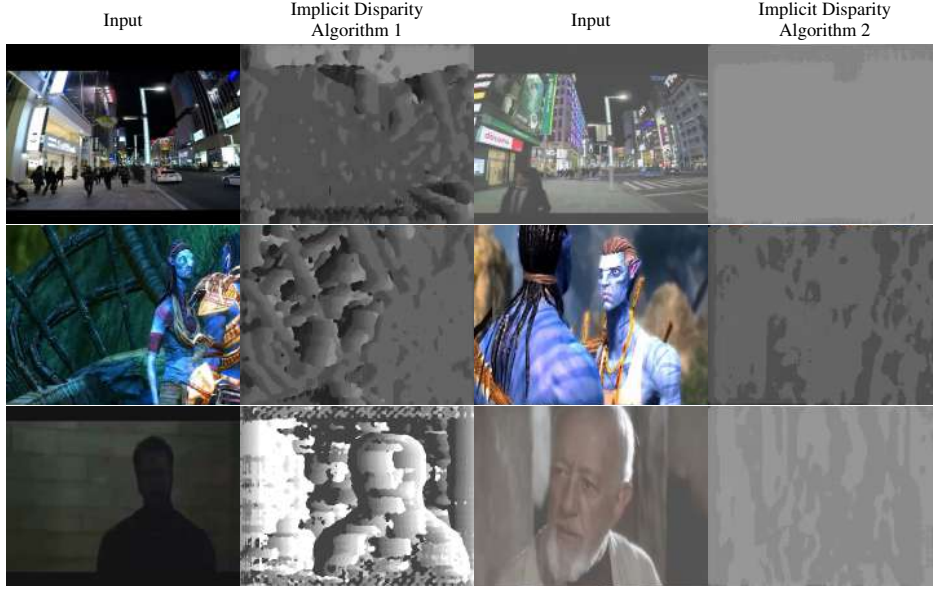


Figure 7: Visual demonstration of the implicit disparity output for different masking strategies. The implicit disparity map contains multiple channels and we apply *argmax* to obtain the visual output.

in frames with complex feature patterns, as illustrated in Figure 8. We suspect that these edge cases could be mitigated with a larger training dataset that includes greater internal variance, allowing the model to better handle such intricate scenarios.



Figure 8: Visual demonstration of the failed edge cases of context fusion.

### B.3 FLOW-GUIDED FEATURE PROPAGATION

Video feature propagation and deformation have shown their effectiveness for many video-based tasks Xue et al. (2019); Wang et al. (2019b); Haris et al. (2019). The flow-guided deformation concept is particularly suitable for the stereo conversion scenario as the pixel shifting nature according to the disparities. Similar to E<sup>2</sup>FGVI Li et al. (2022) and ProPainter Zhou et al. (2023), we use a similar design of flow-guided feature propagation module, that features bi-directional optical flow-guided deformable alignments that built on top of deformable convolution networks (DCN) Dai et al. (2017); Zhu et al. (2019).

Given extracted features  $\{E_t | t = 1 \dots T\}$  from a feature encoder where  $T$  is the total number of frames. Under the context of stereo conversion, the forward flow  $F_{t \rightarrow t+1}$  helps to track the movement of occluded regions from frame  $t$  to frame  $t + 1$ . When the pixels within the occluded areas of frame  $t$  are found in the valid regions of frame  $t + 1$ , this information can be utilized effectively by warping the backward propagation feature  $\hat{E}_b^{t+1}$  from frame  $t + 1$  back to frame  $t$ , guided by the forward flow  $F_{t \rightarrow t+1}$ . On top of E<sup>2</sup>FGVI’s approach, we include flow validation maps  $M_{t+1 \rightarrow t}$  by consistency check introduced by ProPainter. The consistency check compares the forward and backward optical flows to ensure the correctness of the used optical flows. Similar to Equation (1),

the consistency error is computed as follows:

$$\mathcal{E}_{t \rightarrow t+1}(p) = \|F_{t \rightarrow t+1}(p) + F_{t+1 \rightarrow t}(p + F_{t \rightarrow t+1}(p))\|_2^2, \quad (4)$$

where  $p$  is pixel positions of the frame. Then the flow deformation offsets  $\tilde{o}_{t \rightarrow t+1}$  are computed with the DCN network, where a concatenation of the forward flow  $F_{t \rightarrow t+1}$ , backward propagation feature  $\hat{E}_b^{t+1}$ , warped backward feature  $\mathcal{W}(\hat{E}_b^{t+1}, F_{t \rightarrow t+1})$ , and flow validation maps  $M_{t+1 \rightarrow t}$  is used as the condition, where  $\mathcal{W}$  is warping operation. The flow-guided alignment propagation is then:

$$\hat{E}_b^t = \mathcal{R}(\mathcal{D}(\hat{E}_b^{t+1}; F_{t \rightarrow t+1} + \tilde{o}_{t \rightarrow t+1}), f_t), \quad (5)$$

where  $\mathcal{D}(\cdot)$  is the deformable convolution layers and  $\mathcal{R}(\cdot)$  fuses the aligned and current features.

However, we found the disparity cannot be learned with those flow-guided propagation modules. We suspect the feature map deformation and alignment can break the internal disparity features, resulting in a failed learning of the implicit disparity maps.

#### B.4 DIFFERENT BACKBONES FOR THE DEPTH BRANCH

We provide additional results in table 4. We experimented with MiDaS instead of DepthAnything. The results indicate that different depth estimation backbones do not affect the performance of our proposed method.

	Depth Backbone	L1 ↓	SSIM ↑	PSNR ↑
Ours w/o context fusion	MiDaS	0.0590	0.6014	21.6572
Ours w/o context fusion	DepthAnything	0.0588	0.5959	21.6649

Table 4: Additional results. The best results are highlighted in green.